Tensors

Yubo Tao

June 4, 2019

Outline

- Examples of Applications of Tensors
- Tensors
  - Matrices, SVD, QR, LSI, NMF, Alternating Minimization
  - Rotation problem, Tensors, The rank of tensors
  - Differences between matrices and tensors
  - Tensor decompositions
- Solution Approaches
  - Tucker tensors
  - Jennrich's algorithm (PARFAC, Kruskal tensors)
- Conclusions

Reference:
Tensors https://web.stanford.edu/class/cs168/l/l10.pdf
Mining and Forecasting of Big Time-series Data, SIGMOD 2015 Tutorial
Algorithmic Aspects of Machine Learning, Ankur Moitra

What is a Tensor?

- A tensor is just like a matrix, but with more dimensions
- Definition: A $n_1 \times n_2 \times \cdots \times n_k$ *k*-tensor is a set of $n_1 \cdot n_2 \cdot \cdots \cdot n_k$ numbers, which one interprets as being arranged in a *k*-dimensional hypercube. Given such a *k*-tensor, *A*, we can refer to a specific element via $A_{i_1, i_2, \cdots i_k}$
  - A 2-tensor is simply a matrix
  - A 3-tensor is a stack of matrices, $A_{i,j,k}$ refer to the *i*,*j*th entry of the *k*th matrix
- In most computer science applications involving data, the above definition of tensors suffices, but ...

Examples of Tensors

- 2-tensors (matrices)
  - Graph – social network

	John	Peter	Mary	Nick	…
John	0	11	22	55	…
Peter	5	0	6	7	…
Mary	…	…	…	…	…
Nick	…	…	…	…	…
…	…	…	…	…	…

  - Cloud of n-d points

	chol#	blood#	age	..	…
John	13	11	22	55	…
Peter	5	4	6	7	…
Mary	…	…	…	…	…
Nick	…	…	…	…	…
…	…	…	…	…	…

Examples of Tensors

- 2-tensors (matrices)
  - Market basket

	milk	bread	choc.	wine	...
John	13	11	22	55	...
Peter	5	4	6	7	...
Mary	...	...	...	...	...
Nick	...	...	...	...	...
...	...	...	...	...	...

  - Documents and terms

	data	mining	classif.	tree	...
Paper#1	13	11	22	55	...
Paper#2	5	4	6	7	...
Paper#3	...	...	...	...	...
Paper#4	...	...	...	...	...
...	...	...	...	...	...

Examples of Tensors

- 2-tensors (matrices)
  - Authors and terms

	data	mining	classif.	tree	...
John	13	11	22	55	...
Peter	5	4	6	7	...
Mary	...	...	...	...	...
Nick	...	...	...	...	...
...	...	...	...	...	...

  - Sensor-ids and time-ticks

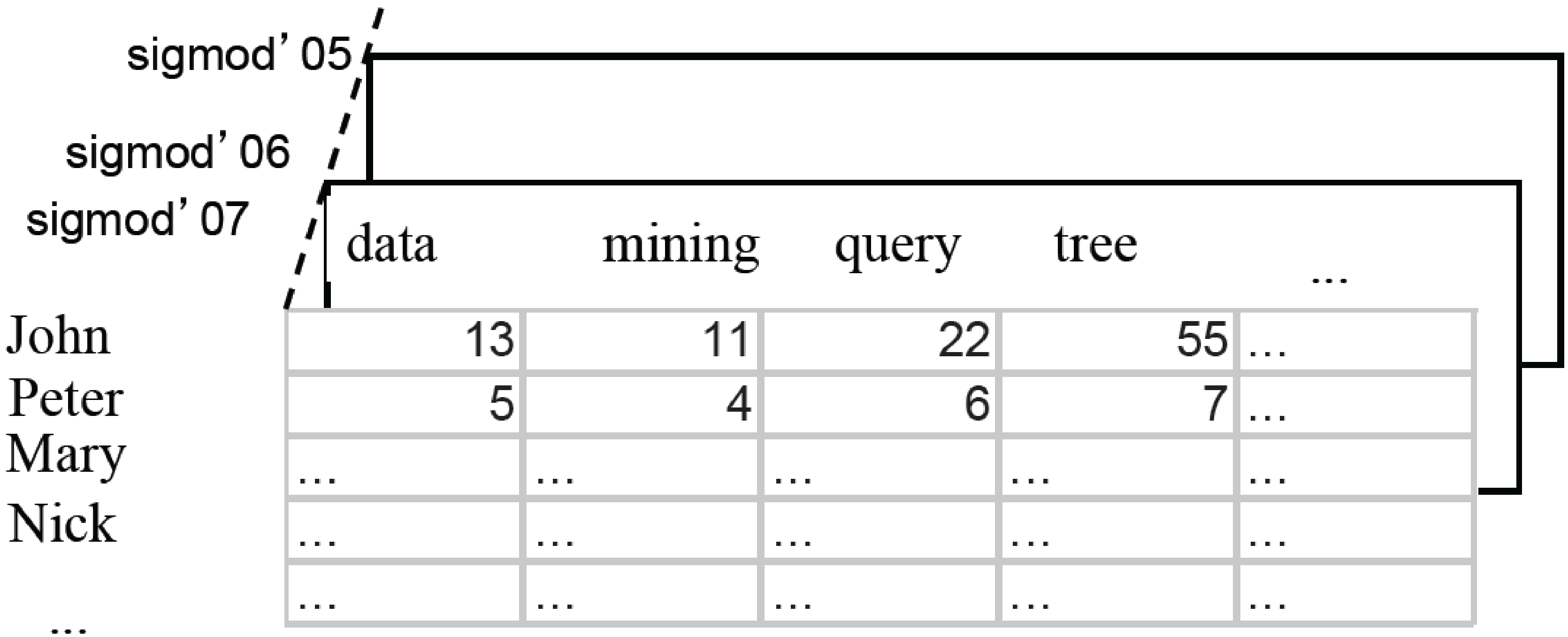
	temp1	temp2	humid.	pressure	...
t=1	13	11	22	55	...
t=2	5	4	6	7	...
t=3	...	...	...	...	...
t=4	...	...	...	...	...
...	...	...	...	...	...

Examples of Tensors

- 2-tensors (matrices)
- k-grams
  - Given a body of text, and some ordering of the set of words, we can associate a k-tensor, A, defined by setting entry $A_{i_1,\cdots i_k}$ equal to the number of times the sequence of words $w_{i_1}, w_{i_2}, \ldots, w_{i_k}$ occurs in the text
- The Moment Tensor
  - Suppose we have some data $s_1, s_2, \ldots, s_n$ representing independent draws from some high-dimensional distribution in R^d. The covariance matrix of this data is represented by a $d \times d$ matrix, whose *i*,*j*th entry is the empirical estimate of $\mathbb{E}[(X_i - \mathbb{E}[X_i])(X_j - \mathbb{E}[X_j])]$. A $d \times d \times d$ tensor $M_{i,j,k} = \frac{1}{n}\sum_{\ell=1}^{n}(s_{\ell_i} - m_i)(s_{\ell_j} - m_j)(s_{\ell_k} - m_k)$

Examples of Tensors

- 2-tensors (matrices)
- k-grams
- The Moment tensor
- Author-terms-year tensor

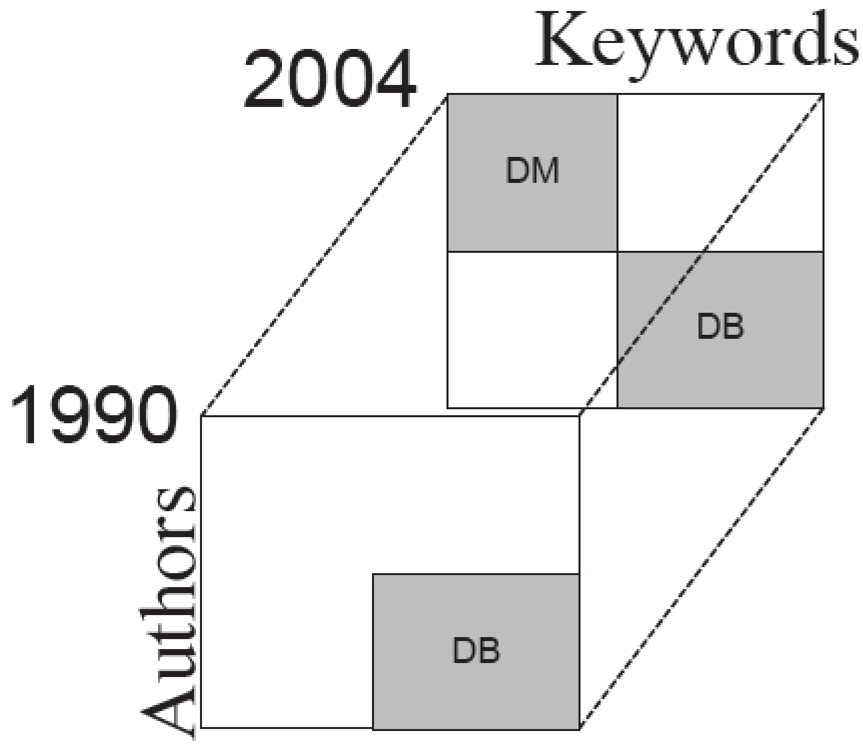


Applications

- Social network analysis
- Web graph mining
- Time-series analysis
- Temporal-spatial analysis
- Document topic modeling
- Computational biology
- ……

Social Network Analysis

- Traditionally, people focus on static networks and fine community structures
- Monitor the change of the community structure over time



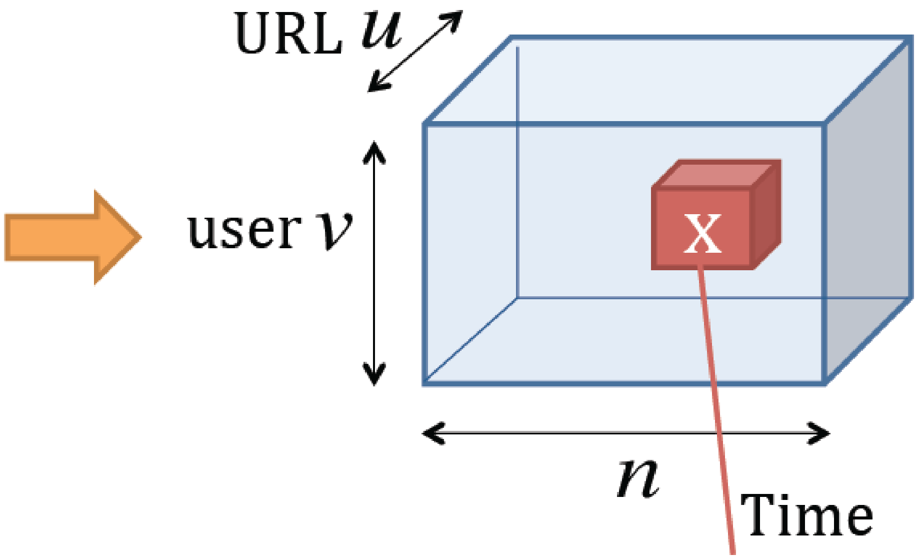
Web Graph Mining

- How to order the importance of web pages?
  - Kleinberg’s algorithm HITS
  - PageRank
  - Tensor extension on HITS (TOPHITS)
    - Context-sensitive hypergraph analysis

Time-Series Analysis

- Time-stamped events
  - *e.g., web clicks*

Time	URL	User
08-01-12:00	CNN.com	Smith
08-02-15:00	YouTube.com	Brown
08-02-19:00	CNET.com	Smith
08-03-11:00	CNN.com	Johnson
...	...	...



Represent as
M^{th} order tensor (M=3)

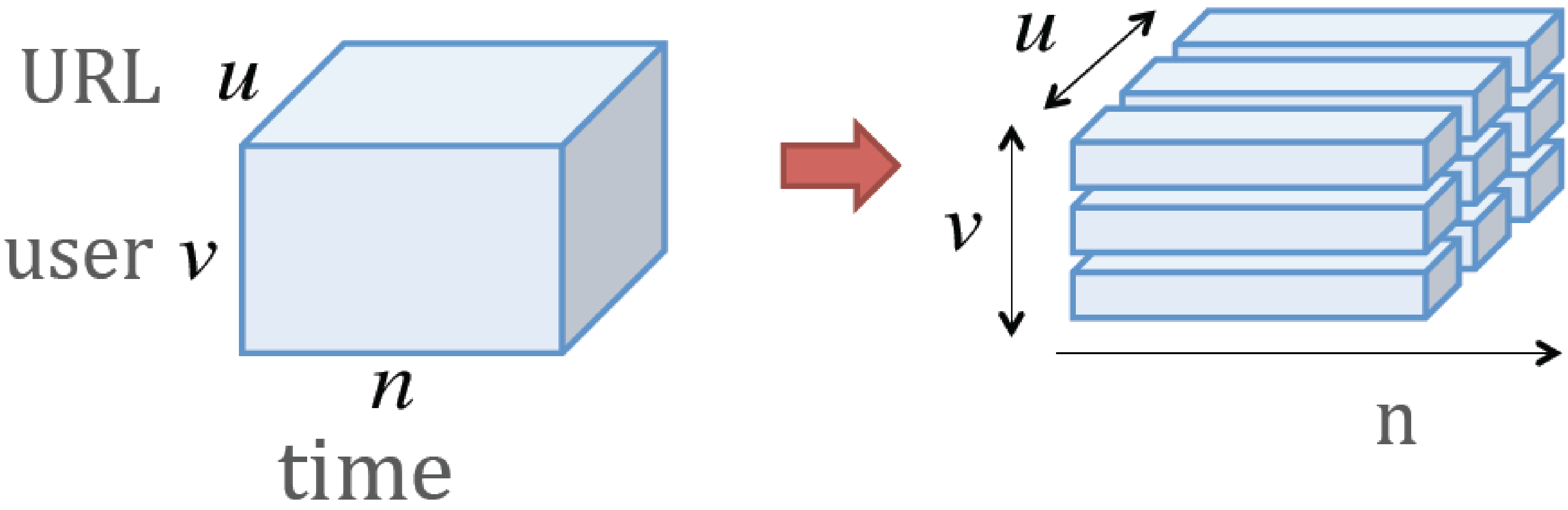
$$\mathcal{X} \in \mathbb{N}^{u \times v \times n}$$

Element x: # of events

e.g., 'Smith', 'CNN.com', 'Aug 1, 10pm'; 21 times

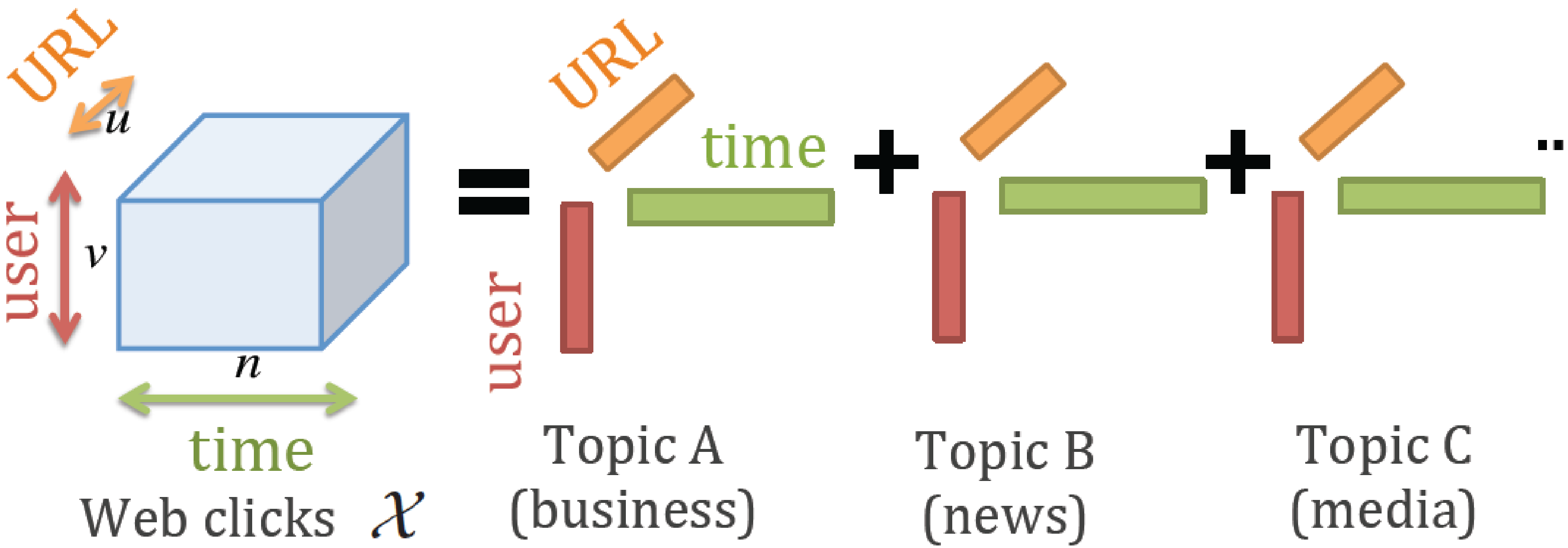
Time-Series Analysis

- Individual-sequence mining
  - Create a set of (u * v) sequences of length (n)
  - Apply the mining algorithm for each sequence

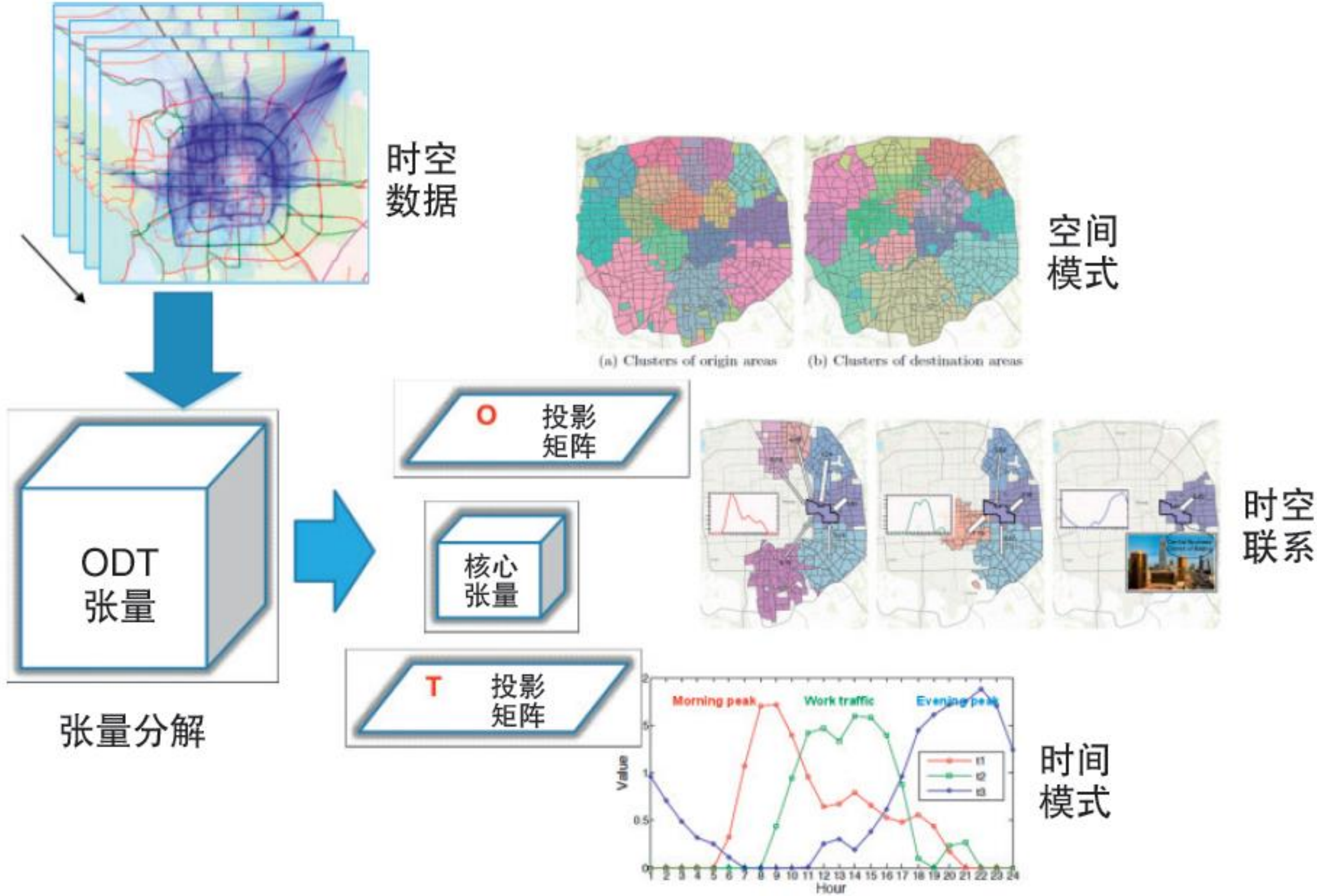


Time-Series Analysis

- Multi-aspect time-series analysis



Temporal-Spatial Analysis



Document Topic Modeling

- Each document is a distribution on topics
- Each topic is a distribution on words

Parceling Out a Nest Egg, Without Emptying It

By PAUL SULLIVAN

What clients often forget are fixed costs — homes, cars, insurance — that must come down but take time to reduce, she said. Beyond that is her clients' skittish approach to risk; putting all of their money in cash may make them feel safe, she said, but it probably will not support the lifestyle they want for decades.

A generational disconnect is at work here: most people plan to retire at 65, the retirement age established for Social Security in 1935, when the average life expectancy was 61. Today the average is over 80 for men and women with a college degree.

So the $5.12 million gift exemption — created in a compromise between President Obama and Congress in 2010 — presents the well-off with a decision laden with short- and long-term consequences. How much should they give heirs now — and thus avoid giving the government in estate taxes later — while maintaining their lifestyle over a probably longer

Personal Finance: (money, 0.15), (retire, 0.10), (risk, 0.03) ...

Politics: (President Obama, 0.10), (congress, 0.08), (government, 0.07), ...

Outline

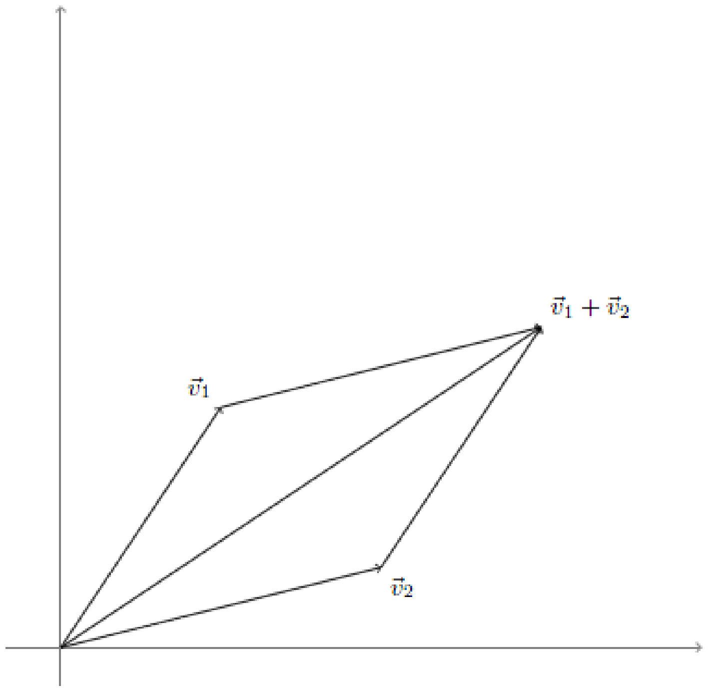
- Examples of Applications of Tensors
- Tensors
  - Matrices, SVD, QR, LSI, NMF, Alternating Minimization
  - Rotation problem, Tensors, the rank of tensors
  - Differences between matrices and tensors
  - Tensor decompositions
- Solution Approaches
  - Tucker tensors
  - Jennrich’s algorithm (PARFAC, Kruskal tensors)
- Conclusions

Vector Spaces

- R^n
  - A set
  - The element p $= (x_1, x_2, x_3, \ldots, x_n)$
- R^n + vector addition + scalar multiplication
  - A vector space
  - The element

$$v = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$$

  - Vector addition $v_1 + v_2$
  - Scalar multiplication cv



Linear Maps

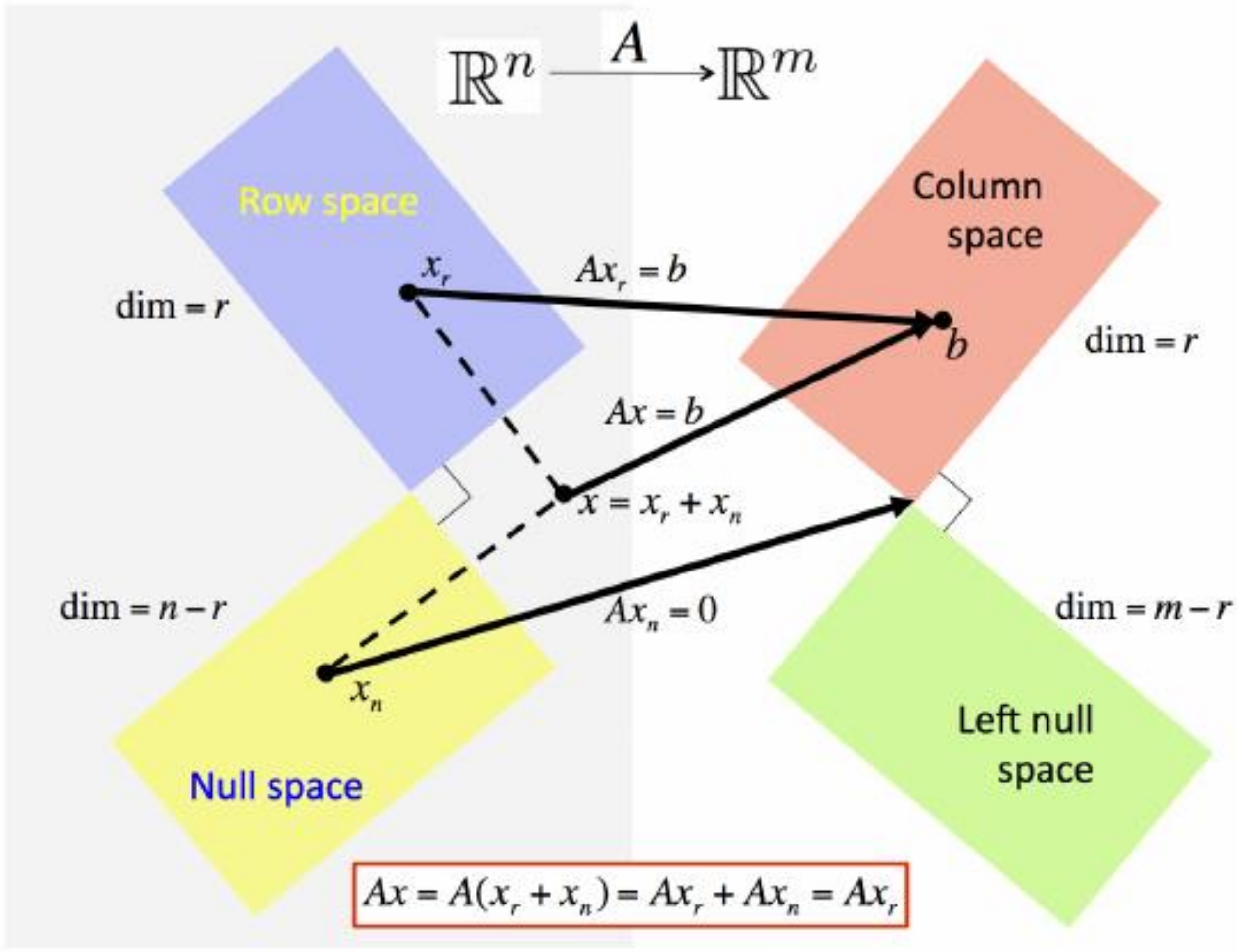
- Definition: Let V and W be two vector spaces. A function $L: V \to W$ is a linear map if
  - For all $v, w \in V$, we have $L(v + w) = L(v) + L(w)$
  - For all $v \in V$ and $a \in R$, we have $L(cv) = cL(v)$
- Matrix of a linear map
  - Matrices record where basis vectors go
- Example: Let V be a vector space with basis (v_1, v_2, v_3) and W be a vector space with basis (w_1, w_2). Suppose there is a linear map $L: V \to W$ for which $L(v_1) = 3w_1 + 2w_2$, $L(v_2) = 3w_1 - 2w_2$, $L(v_3) = w_1 + w_2$, $A_L = \begin{bmatrix} 3 & 3 & 1 \\ 2 & -2 & 1 \end{bmatrix}$. $x = (2,3,-4)$, $L(x)$=?

Matrix of A Linear Map

- $A \in R^{m \times n}$
- The column space of A is $C(A) = \{y \,|y = Ax, x \in R^n\}$ in R^m
  - $C(A)$ is a subspace R^m
  - $Ax = b$ has a solution if and only if $b \in C(A)$
- The null space of A is $N(A) = \{x \mid Ax = 0\}$ in R^n
  - The kernel of a linear map
- The row space of A is $R(A) = \{y \,|y = A^T x, x \in R^m\}$ in R^n
  - $R(A) \perp N(A)$
- The left null space of A is $N(A^T) = \{x \mid x^T A, x \in R^n\}$ in R^m

Matrix of A Linear Map

- $A \in R^{m \times n}$

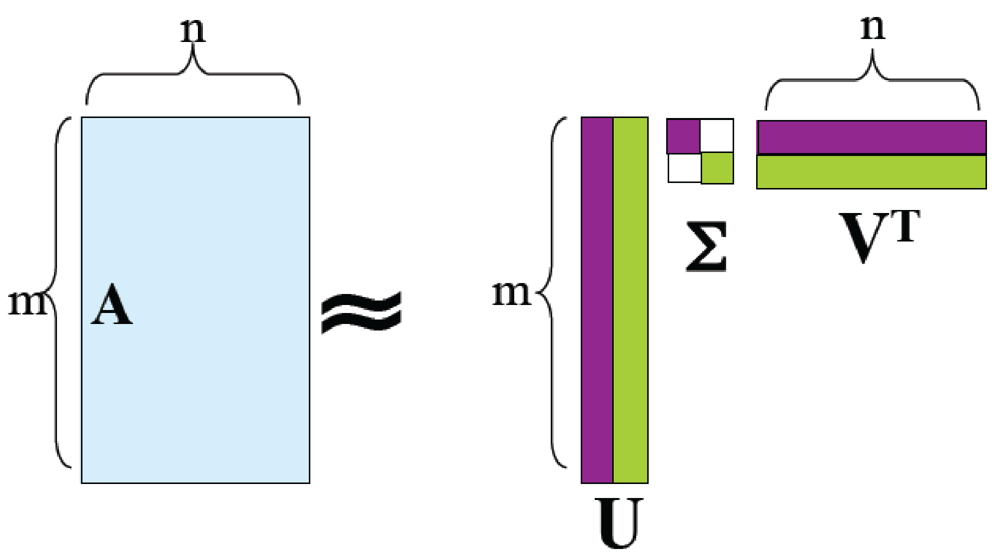


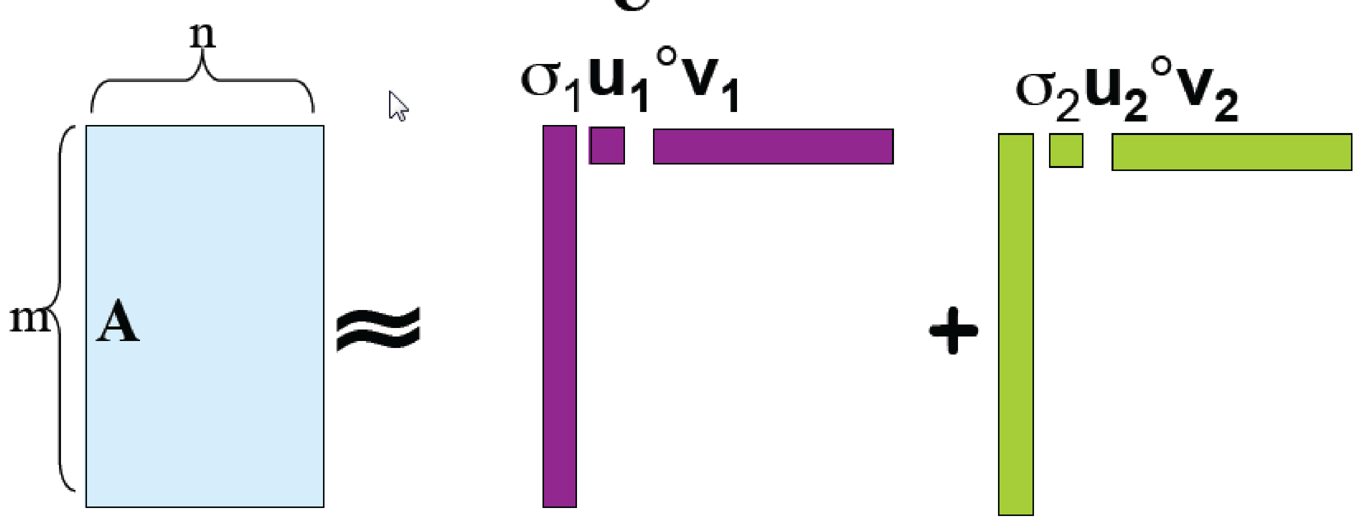
Matrix of A Linear Map

- Let us assume that $A \in R^{m\times n}$ $(m > n$, overdetermined systems) has linearly independent columns and we wish to solve $Ax \approx b$ (normal equation $A^T Ax = A^T b$)
  - The orthogonal projection of b onto $C(A)$ is given by
  $$\hat{b} = A(A^T A)^{-1} A^T b$$
  - The vector (b-$\hat{b}$) is the component of b orthogonal to $C(A)$
- Example: given $a, b \in R^n$
  - Component of b in the direction of a:
    - $u = \frac{a^T b}{a^T a} a = a(a^T a)^{-1} a^T b$
  - Matrix that project onto Span({a}): $a(a^T a)^{-1} a^T$
  - Component of b orthogonal to a:
    - $w = b - a(a^T a)^{-1} a^T b = (I - a(a^T a)^{-1} a^T)\text{b}$
  - Matrix that projects onto Span({a})$^\perp$: $I - a(a^T a)^{-1} a^T$

Singular Value Decomposition

$$\mathbf{A} \approx \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i$$



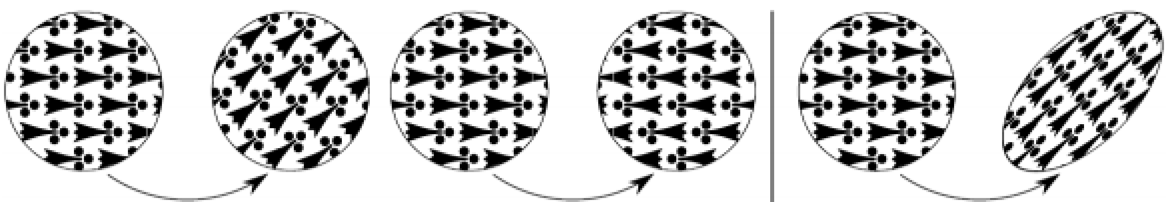


Singular Value Decomposition

- The Frobenius norm of a matrix *M* is $\|M\|_F = \sqrt{\sum_{i,j} M_{i,j}^2}$
  - If $M = \sum_{i=1}^{r} u_i \sigma_i v_i^T$, $\|M\|_F = \sqrt{\sum \sigma_i^2}$
- Eckart-Young: The best rank k approximation to M in Frobenius norm is attained by $B = \sum_{i=1}^{k} u_i \sigma_i v_i^T$, and its error is $\|M - B\|_F = \sqrt{\sum_{i=k+1}^{r} \sigma_i^2}$
  - Best rank-k approximation in L2
- The operator norm of a matrix M is $\|M\|_2 = \max_{|v|=1} \|Mv\|_2$
  - If $M = \sum_{i=1}^{r} u_i \sigma_i v_i^T$, $||M||_2 = \sigma_1$ *(the largest singular value)*
- Approach
  - Reduce M to bidiagonal form using Householder reflections, and compute the singular value decomposition using the QR algorithm

QR Factorization

- Orthogonal matrix
  - A set of vector $\{v_1, \ldots, v_n\}$ is orthonormal if $\|v_i\| = 1$ for all i and $v_i \cdot v_j = 0$ for all $i \neq j$. A square matrix whose colums are orthonormal is called an orthogonal matrix
  - $Q = [v_1\ v_2 \cdots v_n]$, $Q^T Q = I_{n\times n}$, $\|Qx\| =?$, $(Qx)^T \cdot (Qy) =?$



(a) Isometric (b) Not isometric

- Column space invariance
  - For any $A \in R^{m\times n}$ abd invertible $B \in R^{n\times n}$, col *A* = col *AB*
  - Invertible column operations do not affect column space

QR Factorization

- Least-Squares
  - $A^TAx = A^Tb \leftrightarrow \min_x \|Ax - b\|$
  - $x = (A^TA)^{-1}A^Tb$, potential problems in this?
    - cond $A^TA \approx (\mathrm{cond}\ A)^2$
  - Project b onto the column space of A
  - Apply colum operations to A until it is orthogonal; then solve least-squares on the resulting orthogonal Q
- $A = QR$
  - Q orthogonal
  - R upper triangular
- $A^TAx = A^Tb$, $A = QR$, $\rightarrow x = R^{-1}Q^Tb$
  - Didn't neet to comput A^TA or $(A^TA)^{-1}$

Orthonormal Projection

- Suppose $a_1, \ldots, a_n$ are orthonormal

$$\|c_1 a_1 + c_2 a_2 + \cdots + c_n a_n - b\|_2^2$$

$$= \sum_{i=i}^{n} (c_i^2 - 2c_i b \cdot a_i) + \|b\|_2^2$$

$$c_i = b \cdot a_i$$

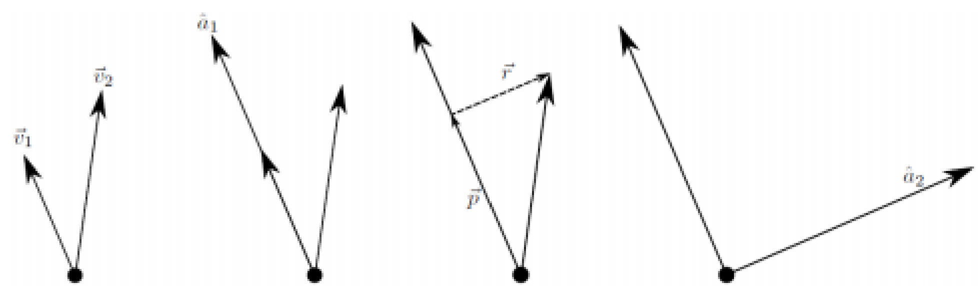
- Gram-Schmidt orthogonalization

To orthogonalize $\vec{v}_1, \ldots, \vec{v}_k$:

1. $\hat{a}_1 \equiv \frac{\vec{v}_1}{\|\vec{v}_1\|}$.
2. For i from 2 to k,

   2.1 $\vec{p}_i \equiv \text{proj}_{\text{span }\{\hat{a}_1, \cdots, \hat{a}_{i-1}\}} \vec{v}_i$.

   2.2 $\hat{a}_i \equiv \frac{\vec{v}_i - \vec{p}_i}{\|\vec{v}_i - \vec{p}_i\|}$.



(a) Input (b) Rescaling (c) Projection (d) Normalization

$$\text{proj}_{\text{span }\{\hat{a}_1, \cdots, \hat{a}_k\}} \vec{b} = (\hat{a}_1 \cdot \vec{b})\hat{a}_1 + \cdots + (\hat{a}_k \cdot \vec{b})\hat{a}_k$$

Applications to Text Analysis

- Latent semantic indexing
  - Document-term matrix

$$M \approx U^{(k)} \Sigma^{(k)} V^{(k)T}$$

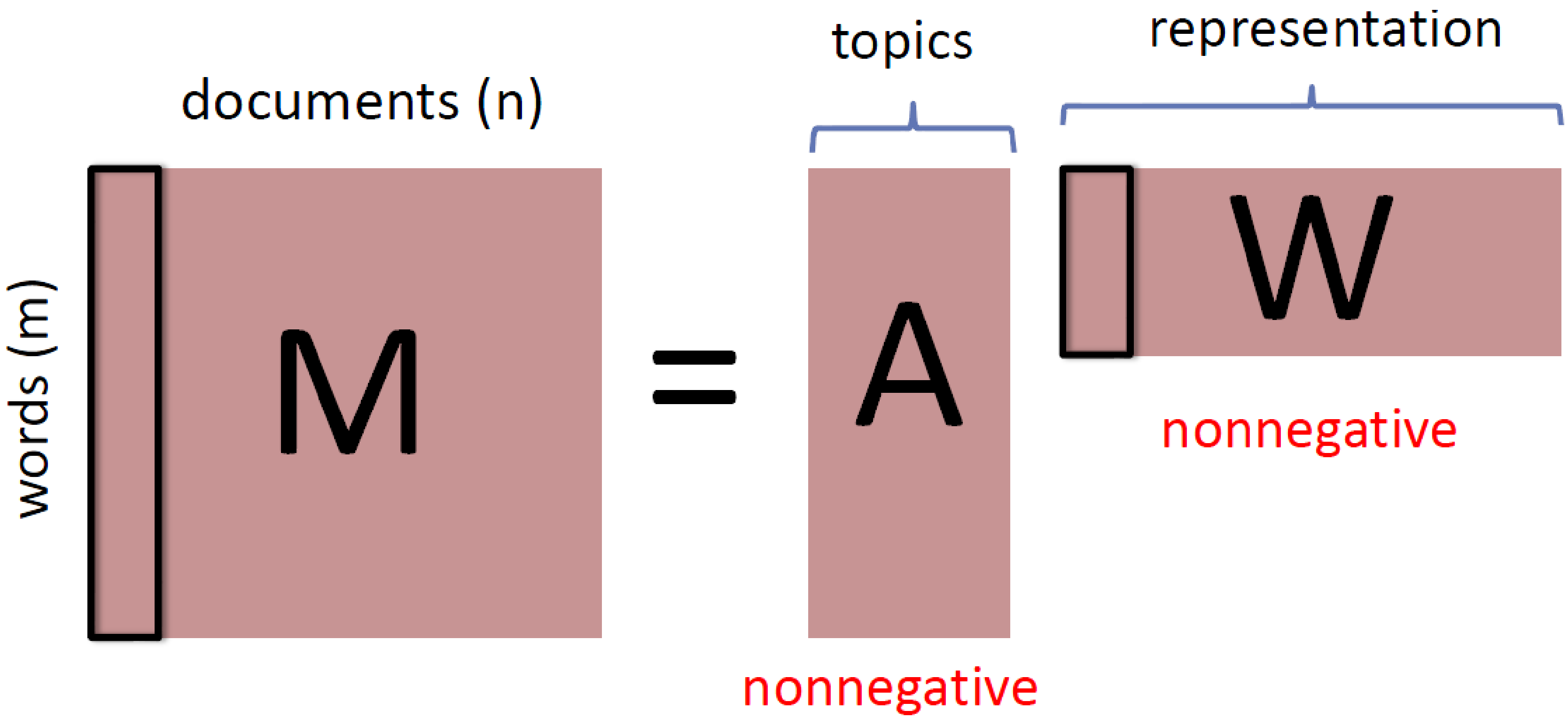
  - Similarity between document *i* and document *j* $\langle M_i, M_j \rangle$
  - Word similarity → topic similarity

$$\langle M_i^T U^{(k)}, M_j^T U^{(k)} \rangle$$

- Limitations
  - Topics are orthonormal
  - Topics contain negative values

Nonnegative Matrix Factorization

- E.g. “personal finance”, (0.15, money), (0.10, retire), (0.03, risk), …



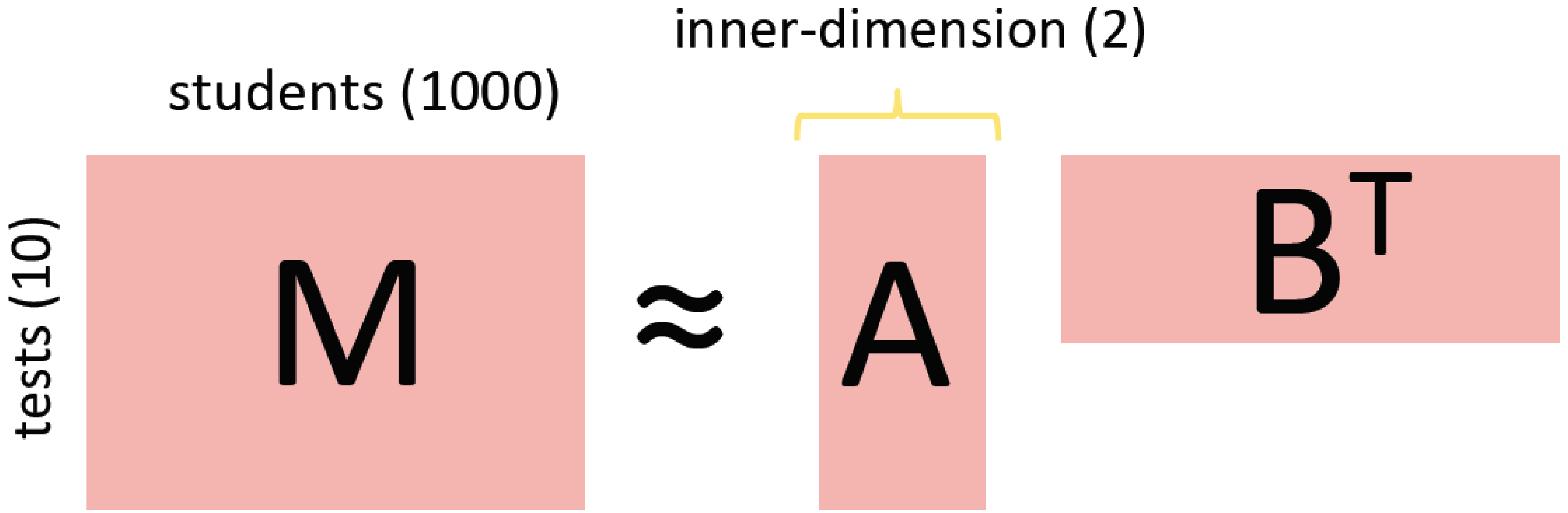
- We can assume columns of A, W sum to one
  - A’ = AD, W’ = D^{-1}W

Nonnegative Matrix Factorization

- This optimization problem is non-convex and NP-hard [Vavasis'09], $(nm)^{r^2}$ [Arora et al. STOC'12]
- Alternating minimization (local search)
  - Set A fixed, compute the nonnegative W that minimizes $||M - AW||_F$ (convex)
  - Set W fixed, compute the nonnegative A that minimizes $||M - AW||_F$ (convex)
- Local search
  - Known to fail on worst-case inputs (Converge, but not necessarily to the optimal solution)
  - Highly sensitive to cost-function, update procedure, regularization

Spearman's Hypothesis

- Charles Spearman (1904): There are two types of intelligence, eductive and reproductive
  - Eductive (adj): the ability to make sense out of complexity
  - Reproductive (adj): the ability to store and reproduce information
- To test this theory, he invented Factor Analysis



Spearman's Hypothesis

Given: $$M = \sum a_i \otimes b_i$$

$$= \underbrace{A \quad B^T}_{\text{“correct” factors}} = \underbrace{AR \quad R^{-1}B^T}_{\text{alternative factorization}}$$

- When can we recover the factors a_i and b_i uniquely
- Claim: The factor $\{a_i\}$ and $\{b_i\}$ are not determined uniquely unless we impose additional conditions on them
  - E.g. if $\{a_i\}$ and $\{b_i\}$ are orthogonal, or rank(M) = 1
- This is called the rotation problem, and is a major issue in factor analysis and motivates the study of tensor methods …

Multilinear Forms

- Definition: Let V be a vector space and let $k \geq 1$. A map $T\colon V^k \to R$ is called a *k*-linear form if for each $i = 1, \ldots, k$ the map $T_i\colon V \to R$ given by the rule $T_i(x) = T(v_1, \ldots, v_{i-1}, x, v_{i+1}, \ldots, v_k)$ is linear
  - For $k > 1$ we call a *k*-linear form a multilinear form
- Let V be a finite dimensional vector space and let $k > 1$ be an integer. Then collection $\vartheta^k(V)$ of k-linear forms $V^k \to R$ is a vector space

Multilinear Forms

- Definition: Let T: $V^{k_1} \to R$ and S: $V^{k_2} \to R$ be multilinear forms. Then we define their tensor product by $T \otimes S$: $V^{k_1+k_2} \to R$ by the rule

$$(T \otimes S)\left(v_1, \dots, v_{k_1+k_2}\right)$$
$$= \mathrm{T}\left(v_1, \dots, v_{k_1}\right)\mathrm{S}(v_{k_1+1}, \dots, v_{k_1+k_2})$$

Higher Order Derivatives

- The (*k+1*)st order derivative of a function $f\colon R^n \to \mathrm{R}$ at a point p is a (*k+1*)-linear form $D^{k+1}f(p)$, which allows us to approximate changes in the *k*th order derivative
- Definition: Let $f\colon R^n \to \mathrm{R}$ be $k+1$ times differentiable, let $p \in R^n$, and let v_1 , …, $v_{k+1} \in R^n$. Then we recursively define the $(k+1)$-linear map $D^{k+1}f(p)\colon (R^n)^{k+1} \to R$ by the rule

$$D^k f(p + v_{k+1})(v_1, \dots, v_{k+1}) = D^k f(p)(v_1, \dots, v_{k+1}) + D^{k+1} f(p)(v_1, \dots, v_{k+1}) + \varepsilon(p)(v_1, \dots, v_{k+1})$$

where $\lim_{v_1,\dots,v_{k+1} \to 0} \frac{\varepsilon(p)(v_1,\dots,v_{k+1})}{\|v_1\| \dots \|v_{k+1}\|} = 0$

Higher Order Derivatives

- Let $f\colon R^n \rightarrow$ R be a continuously k-times differentiable function. Then the k-linear form representing the kth order derivative is a symmetric form
- Example $f(x, y) = x^2 y$, find the third derivative of f at (0, 0)
  - $D^3 f(x, y) = 2dx \otimes dx \otimes dy + 2dx \otimes dy \otimes dx + 2dy \otimes dx \otimes dx$
  - $D^3 f(0,0)(\begin{bmatrix}1\\2\end{bmatrix}, \begin{bmatrix}3\\4\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix})$
  - = 2 * 1 * 3 * 1 + 2 * 1 * 4 * 0 + 2 * 2 * 3 * 0 = 6

Taylor's Theorem

- Let $f\colon R^n \to \mathrm{R}$ be $(k+1)$-times continuously differentiable and choose $p \in R^n$ and $h \in R^n$. Then there exists $\alpha \in [0,1]$ such that

$$f(p+h)$$
$$= f(p) + Df(p)(h) + \frac{1}{2!}D^2 f(p)(h^2) + \frac{1}{3!}D^3 f(p)(h^3)$$
$$+ \cdots + \frac{1}{k!}D^k f(p)\left(h^k\right) + \frac{1}{(k+1)!}D^{k+1} f(p+ah)(h^{k+1})$$

where h^k means $(h, h, \dots, h)$ (repeated k times)

Tensors

- A tensor is just like a matrix, but with more dimensions
- Definition: A $n_1 \times n_2 \times \cdots \times n_k$ *k*-tensor is a set of $n_1 \cdot n_2 \cdot \cdots \cdot n_k$ numbers, which one interprets as being arranged in a *k*-dimensional hypercube. Given such a *k*-tensor, *A*, we can refer to a specific element via $A_{i_1, i_2, \cdots i_k}$
  - A 2-tensor is simply a matrix
  - A 3-tensor is a stack of matrices, $A_{i,j,k}$ refer to the *i*,*j*th entry of the *k*th matrix

The Rank of a Tensor

- The rank of a tensor is defined analogously to the rank of a matrix
- A matrix M has rank r if it can be written as $M = UV^T$, where U has r columns, and V has r columns.
- Letting $u_1, \ldots u_r$, and $v_1, \ldots v_r$ denote these columns

$$M = \sum_{i=1}^{r} u_i v_i^t$$

  - Outer-product
  - A sum of rank 1 matrices

The Rank of a Tensor

- Tensor product and the rank of a tensor

Definition Given vectors $v_1, v_2, \ldots, v_k$, of lengths $n_1, n_2, \ldots, n_k$, the *tensor product* is denoted $v_1 \otimes v_2 \otimes \ldots \otimes v_k$ is the $n_1 \times n_2 \times \ldots \times n_k$ k-tensor A with entry $A_{i_1, i_2, \ldots, i_k} = v_1(i_1) \cdot v_2(i_2) \cdot \ldots \cdot v_k(i_k)$

Example For example, given

$$v_1 = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix}, v_2 = \begin{pmatrix} -1 \\ 1 \end{pmatrix}, v_3 = \begin{pmatrix} 10 \\ 20 \end{pmatrix}, .$$

$v_1 \otimes v_2 \otimes v_3$ is a $3 \times 2 \times 2$ 3-tensor, that can be thought of as a stack of two 3×2 matrices

$$M_1 = \begin{pmatrix} -10 & 10 \\ -20 & 20 \\ -30 & 30 \end{pmatrix}, M_2 = \begin{pmatrix} -20 & 20 \\ -40 & 40 \\ -60 & 60 \end{pmatrix}.$$

Definition A 3-tensor A has rank r if there exists 3 sets of r vectors, $u_1, \ldots, u_r$, $v_1, \ldots, v_r$ and $w_1, \ldots, w_r$ such that

$$A = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$$

Matrices vs. Tensors

- In general, most of what you know about linear algebra for matrices does NOT apply to tensors

For matrices, the best rank-k approximation can be found by iteratively finding the best rank-1 approximation, and then subtracting it off. In other words, for a matrix M, the best rank 1 approximation of M is the same as the best rank 1 approximation of the matrix M_2 defined as the best rank 2 approximation of M. Because of this, if uv^t is the best rank 1 approximation of M, then $rank(M - uv^t) = rank(M - 1)$.

For k-tensors with $k \geq 3$, this is not always the case. If $u \otimes v \otimes w$ is the best rank 1 approximation of 3-tensor A, it is possible that $rank(A - u \otimes v \otimes w) > rank(A)$.

For matrices with entries in $\mathbb{R}$, there is no point in looking for a low-rank decomposition that involves complex numbers, because $rank_{\mathbb{R}}(M) = rank_{\mathbb{C}}(M)$. For k-tensors, with $k \geq 3$, this is not always the case, it can be that the rank over complex vectors is smaller than the rank over real vectors, even if the entries in the tensor are real-valued.

Matrices vs. Tensors

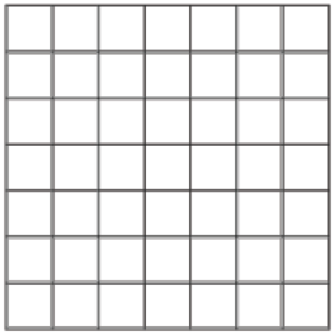
- In general, most of what you know about linear algebra for matrices does NOT apply to tensors

We don't understand tensor rank: for example, with probability 1, if you pick the entries of an $n \times n \times n$ 3-tensor independently at random from the interval $[0, 1]$, the rank will be on the order of n^2, however we don't know how to describe any explicit construction of $n \times n \times n$ tensors whose rank is greater than $n^{1.1}$, for all n.

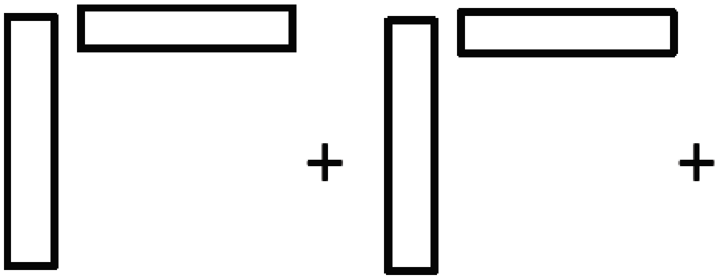
Computing the rank of matrices is easy (e.g. via SVD). Computing the rank of 3-tensors is NP-hard.

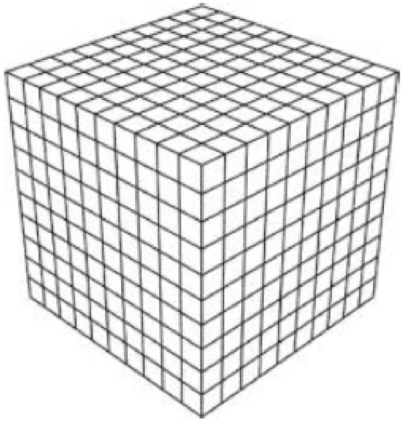
As we will explore in the following section, despite the above point, if the rank of a 3-tensor is sufficiently small, then its rank can be efficiently computed, its low-rank representation is *unique*, and can be efficiently recovered.

Tensor Decompositions



$$M = a_1 \otimes b_1 + a_2 \otimes b_2 + \cdots + a_R \otimes b_R$$

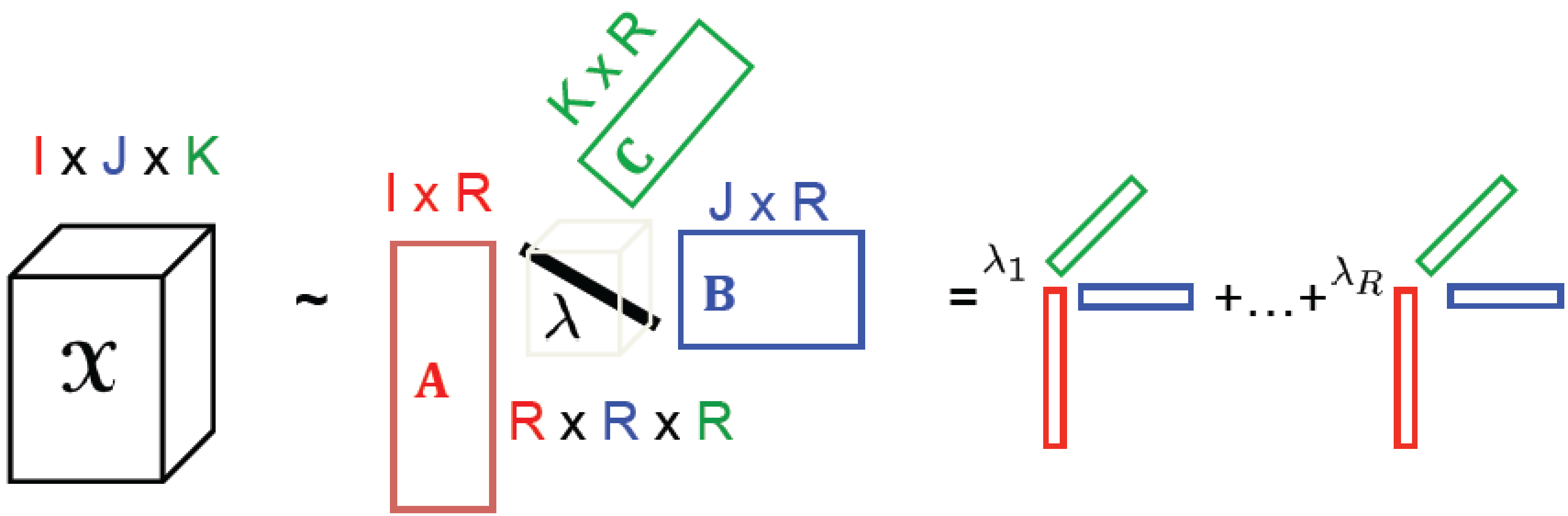




$$T = a_1 \otimes b_1 \otimes c_1 + \cdots + a_R \otimes b_R \otimes c_R$$

(i, j, k) entry of $x \otimes y \otimes z$ is $x(i) \times y(j) \times z(k)$

Tensor Decompositions



$$\mathcal{X} \approx [\![\lambda \, ; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_r \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

Tensor Decompositions

- Rotation problem for matrices
- Once one goes from to 3-tensors, low-rank decompositions end up being essentially unique
- Theorem: Given a 3-tensor A of rank k s.t. there exists three sets of linearly independent vectors, $(u_1, \ldots, u_k)$, $(v_1, \ldots, v_k)$, $(w_1, \ldots, w_k)$, s.t.

$$A = \sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$$

then this rank k decomposition is unique (up to scaling the vectors by constant), and these factors can be efficiently recovered

Tensor Decompositions

Theorem *Consider a tensor*

$$T = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$$

where each set of vectors $\{u_i\}_i$ *and* $\{v_i\}_i$ *are linearly independent, and moreover each pair of vectors in* $\{w_i\}_i$ *are linearly independent too. Then the above decomposition is unique up to rescaling, and there is an efficient algorithm to find it.*

Tensor Decompositions

- Spearman’s Hypothesis
  - Different settings: classical music, video playing, a control setting
  - True model: as above, for every student there are two numbers representing their math/verbal ability, and every test can be regarded as having a math/verbal component; additionally, for each setting, there is some scaling of the math performance resulting from that setting, and a scaling of the verbal performance resulting from that setting
  - Provided the vector of student math abilities is not identical (up to a constant rescaling) to the vector of verbal abilities, and the 2 vectors of math/verbal test components are not identical up to rescaling, and the 2 vectors of math/verbal setting boosts are not identical up to rescaling, then this is the unique factorization of this tensor, and we will be able to recover these exact factors

Outline

- Examples of Applications of Tensors
- Tensors
  - Matrices, SVD, QR, LSI, NMF, Alternating Minimization
  - Rotation problem, Tensors, the rank of tensors
  - Differences between matrices and tensors
  - Tensor decompositions
- Solution Approaches
  - Tucker tensors
  - Jennrich's algorithm (PARFAC, Kruskal tensors)
- Conclusions

Tensor Decompositions

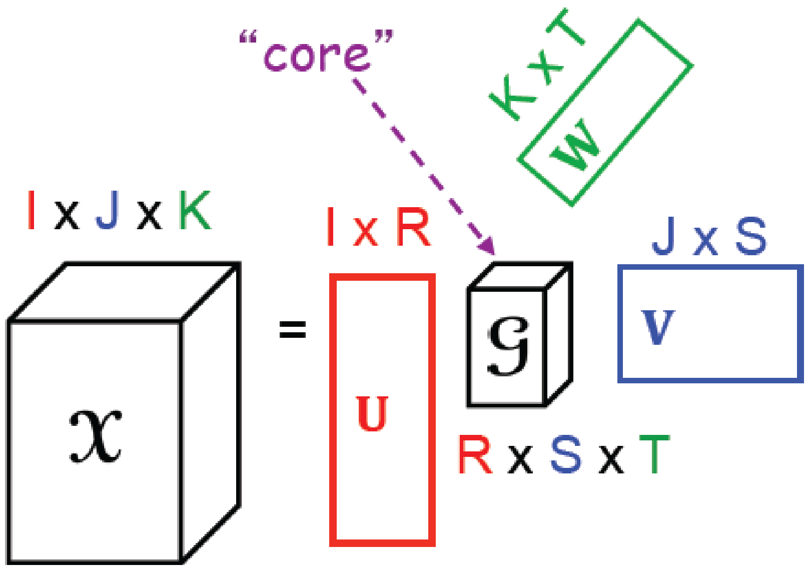
- 2 major types of tensor decompositions
  - Jennrich's Algorithm PARAFAC
  - Tucker
- Both can be solved with “Alternating Lease Squares” (ALS)

Tensor Decompositions

- Tucker Tensor

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}$$
$$= \sum_r \sum_s \sum_t g_{rst}\, \mathbf{u}_r \circ \mathbf{v}_s \circ \mathbf{w}_t$$
$$\equiv [\![\mathcal{G} \,;\, \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$$

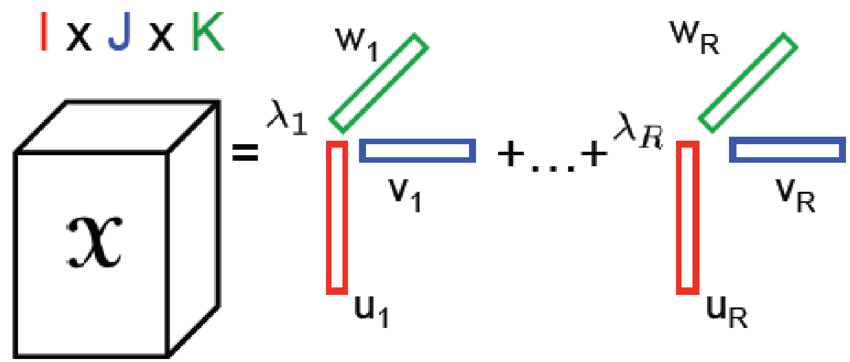
Our Notation



- Kruskal Tensor

$$\mathcal{X} = \sum_r \lambda_r\, \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r$$
$$\equiv [\![\lambda \,;\, \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$$

Our Notation



Tensor Decompositions

- Tucker Tensor

$$\mathcal{X} = \mathcal{G} \times_1 \mathbf{U} \times_2 \mathbf{V} \times_3 \mathbf{W}$$
$$= \sum_r \sum_s \sum_t g_{rst} \, \mathbf{u}_r \circ \mathbf{v}_s \circ \mathbf{w}_t$$
$$\equiv [\![\mathcal{G} \, ; \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$$

In matrix form:

$$\mathbf{X}_{(1)} = \mathbf{U}\mathbf{G}_{(1)}(\mathbf{W} \otimes \mathbf{V})^{\mathsf{T}}$$
$$\mathbf{X}_{(2)} = \mathbf{V}\mathbf{G}_{(2)}(\mathbf{W} \otimes \mathbf{U})^{\mathsf{T}}$$
$$\mathbf{X}_{(3)} = \mathbf{W}\mathbf{G}_{(3)}(\mathbf{V} \otimes \mathbf{U})^{\mathsf{T}}$$

$$\mathrm{vec}(\mathcal{X}) = (\mathbf{W} \otimes \mathbf{V} \otimes \mathbf{U})\mathrm{vec}(\mathcal{G})$$

- Kruskal Tensor

$$\mathcal{X} = \sum_r \lambda_r \, \mathbf{u}_r \circ \mathbf{v}_r \circ \mathbf{w}_r$$
$$\equiv [\![\lambda \, ; \mathbf{U}, \mathbf{V}, \mathbf{W}]\!]$$

In matrix form:

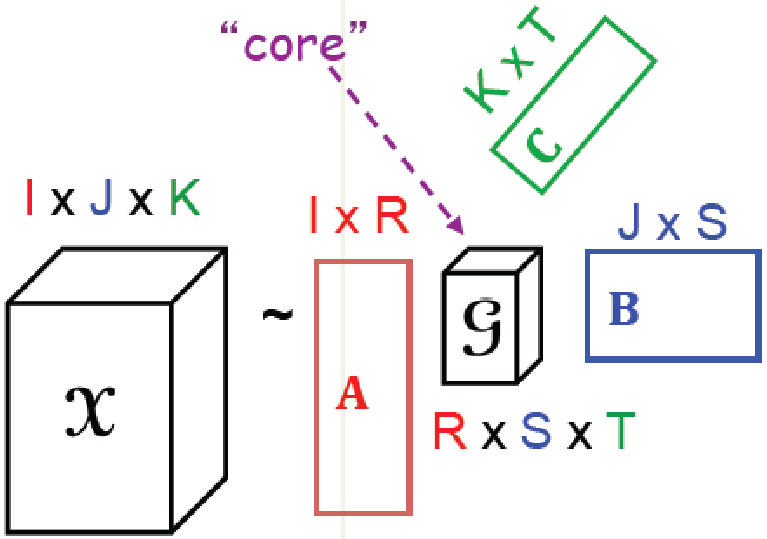
Let $\mathbf{\Lambda} = \mathrm{diag}(\lambda)$

$$\mathbf{X}_{(1)} = \mathbf{U} \, \mathbf{\Lambda} \, (\mathbf{W} \odot \mathbf{V})^{\mathsf{T}}$$
$$\mathbf{X}_{(2)} = \mathbf{V} \, \mathbf{\Lambda} \, (\mathbf{W} \odot \mathbf{U})^{\mathsf{T}}$$
$$\mathbf{X}_{(3)} = \mathbf{W} \, \mathbf{\Lambda} \, (\mathbf{V} \odot \mathbf{U})^{\mathsf{T}}$$

$$\mathrm{vec}(\mathcal{X}) = (\mathbf{W} \odot \mathbf{V} \odot \mathbf{U}) \, \lambda$$

Tucker Decomposition

2D case: co-clustering



- author x keyword x conference
- A: author x author-group
- B: keyword x keyword-group
- C: conf. x conf-group
- $\mathcal{G}$: how groups relate to each other

Tucker Decomposition

$$
\begin{array}{cc}
 & n \\
m & \begin{bmatrix} .05 & .05 & .05 & 0 & 0 & 0 \\ .05 & .05 & .05 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ .04 & .04 & 0 & .04 & .04 & .04 \\ .04 & .04 & .04 & 0 & .04 & .04 \end{bmatrix}
\end{array}
$$

eg, terms x documents

$$
\begin{array}{cc}
 & k \\
m & \begin{bmatrix} .5 & 0 & 0 \\ .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & .5 \\ 0 & 0 & .5 \end{bmatrix}
\end{array}
\begin{array}{cc}
 & l \\
k & \begin{bmatrix} .3 & 0 \\ 0 & .3 \\ .2 & .2 \end{bmatrix}
\end{array}
\begin{array}{cc}
 & n \\
l & \begin{bmatrix} .36 & .36 & .28 & 0 & 0 & 0 \\ 0 & 0 & 0 & .28 & .36 & .36 \end{bmatrix}
\end{array}
=
\begin{bmatrix} .054 & .054 & .042 & 0 & 0 & 0 \\ .054 & .054 & .042 & 0 & 0 & 0 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ .036 & .036 & 028 & .028 & .036 & .036 \\ .036 & .036 & .028 & .028 & .036 & .036 \end{bmatrix}
$$

Tucker Decomposition

med. doc

cs doc

$$\begin{bmatrix} .05 & .05 & .05 & 0 & 0 & 0 \\ .05 & .05 & .05 & 0 & 0 & 0 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ 0 & 0 & 0 & .05 & .05 & .05 \\ .04 & .04 & 0 & .04 & .04 & .04 \\ .04 & .04 & .04 & 0 & .04 & .04 \end{bmatrix}$$

med. terms

cs terms

common terms

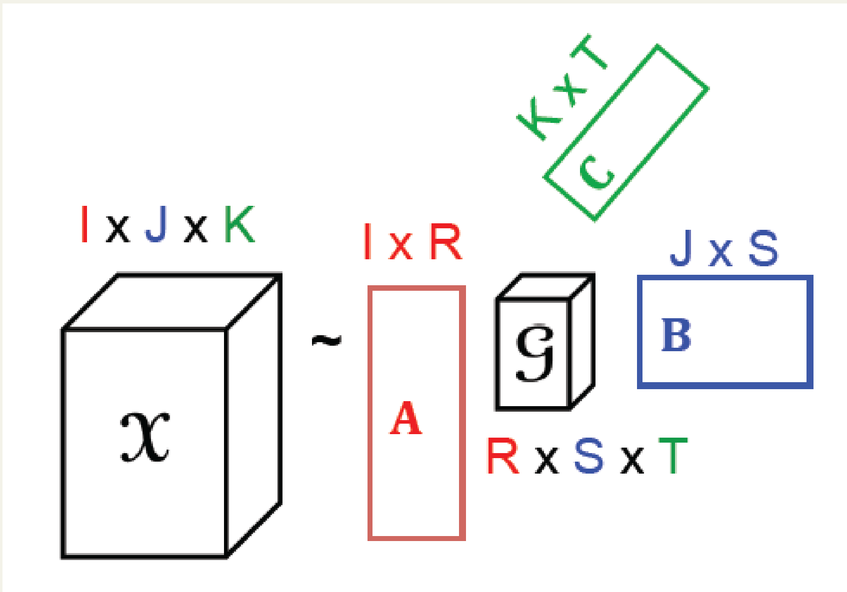
term group x doc. group

$$\begin{bmatrix} .5 & 0 & 0 \\ .5 & 0 & 0 \\ 0 & .5 & 0 \\ 0 & .5 & 0 \\ 0 & 0 & .5 \\ 0 & 0 & .5 \end{bmatrix} \begin{bmatrix} .3 & 0 \\ 0 & .3 \\ .2 & .2 \end{bmatrix} \begin{bmatrix} .36 & .36 & .28 & 0 & 0 & 0 \\ 0 & 0 & 0 & .28 & .36 & .36 \end{bmatrix} = \begin{bmatrix} .054 & .054 & .042 & 0 & 0 & 0 \\ .054 & .054 & .042 & 0 & 0 & 0 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ 0 & 0 & 0 & .042 & .054 & .054 \\ .036 & .036 & 028 & .028 & .036 & .036 \\ .036 & .036 & .028 & .028 & .036 & .036 \end{bmatrix}$$

doc x doc group

term x term-group

Tucker Decomposition



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

Given A, B, C, the optimal core is:

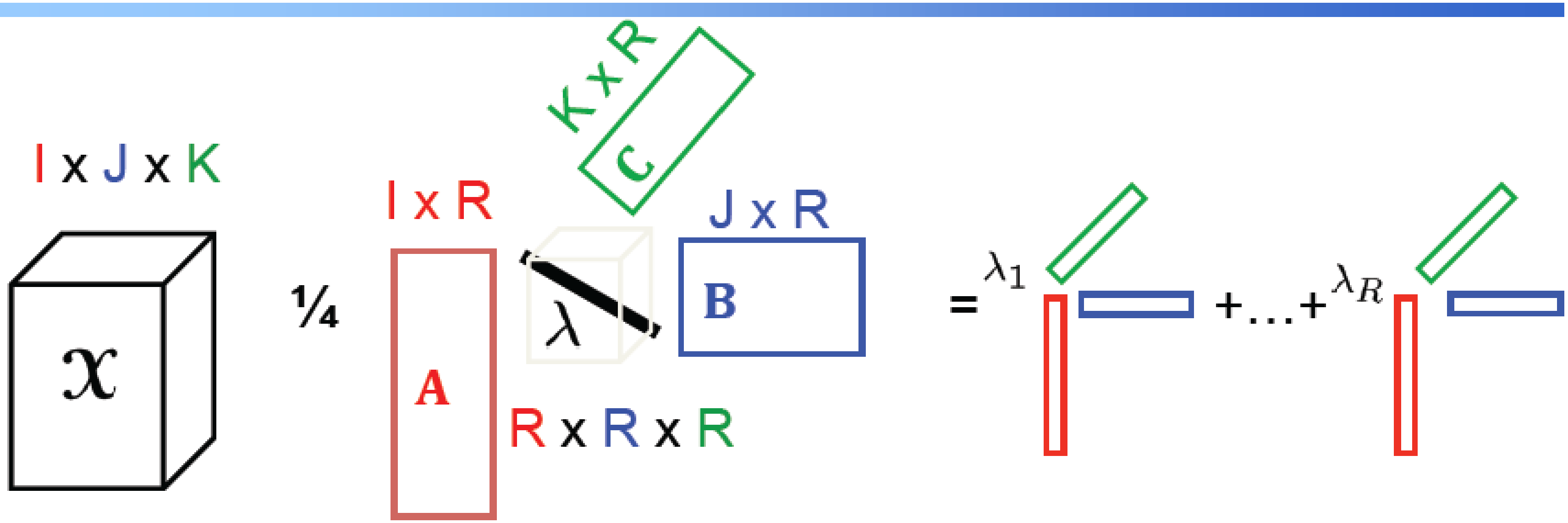
$$\mathcal{G} = [\![\mathcal{X} \,;\, \mathbf{A}^{\dagger}, \mathbf{B}^{\dagger}, \mathbf{C}^{\dagger}]\!]$$

- Proposed by Tucker (1966)
- AKA: Three-mode factor analysis, three-mode PCA, orthogonal array decomposition
- **A**, **B**, and **C** generally assumed to be orthonormal (generally assume they have full column rank)
- $\mathcal{G}$ is <u>not</u> diagonal
- Not unique

Recall the equations for converting a tensor to a matrix

$$\mathbf{X}_{(1)} = \mathbf{A}\mathbf{G}_{(1)}(\mathbf{C} \otimes \mathbf{B})^{\mathsf{T}}$$
$$\mathbf{X}_{(2)} = \mathbf{B}\mathbf{G}_{(2)}(\mathbf{C} \otimes \mathbf{A})^{\mathsf{T}}$$
$$\mathbf{X}_{(3)} = \mathbf{C}\mathbf{G}_{(3)}(\mathbf{B} \otimes \mathbf{A})^{\mathsf{T}}$$
$$\mathrm{vec}(\mathcal{X}) = (\mathbf{C} \otimes \mathbf{B} \otimes \mathbf{A})\mathrm{vec}(\mathcal{G})$$

PARAFAC Decomposition



$$\mathcal{X} \approx [\![\lambda \,; \mathbf{A}, \mathbf{B}, \mathbf{C}]\!] = \sum_r \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r$$

- CANDECOMP = Canonical Decomposition (Carroll & Chang, 1970)
- PARAFAC = Parallel Factors (Harshman, 1970)
- Core is <u>diagonal</u> (specified by the vector λ)
- Columns of **A**, **B**, and **C** are <u>not</u> orthonormal
- If R is <u>*minimal*</u>, then R is called the **rank** of the tensor (Kruskal 1977)
- Can have rank($\mathcal{X}$) > min{I,J,K}

PARAFAC Decomposition

Theorem [Jennrich 1970]: Suppose $\{a_i\}$ and $\{b_i\}$ are linearly independent and no pair of vectors in $\{c_i\}$ is a scalar multiple of each other. Then

$$T = a_1 \otimes b_1 \otimes c_1 + \cdots + a_R \otimes b_R \otimes c_R$$

is unique up to permuting the rank one terms and rescaling the factors.

Equivalently, the rank one factors are **unique**

PARAFAC Decomposition

Tensor Decomposition

Given $n \times n \times p$ tensor $A = \sum_{i=1}^{k} u_i \otimes v_i \otimes w_i$, with $(u_1, \ldots, u_k)$, and $(v_1, \ldots, v_k)$, and $(w_1, \ldots, w_k)$ linearly independent, the following algorithm will output the lists of u's,v's, and w's.

- Choose random unit vectors $a, b \in \mathbb{R}^p$.
- Define the $n \times n$ matrices A_a, A_b, where A_a is defined as follows: consider A as consisting of a stack of p $n \times n$ matrices. Let A_a be the weighted sum of these p matrices, where the weight given to the ith matrix is $a(i)$–namely the ith element of vector a.
- Compute the eigen-decompositions of $A_a A_b^{-1} = QSQ^{-1}$, and $A_a^{-1} A_b = Y^{-1} T Y^t$.
- We will show that with probability 1, the entries of diagonal matrix S will be unique, and will be inverses of the entries of diagonal matrix T. The vectors $u_1, \ldots, u_k$ are the columns of Q corresponding to nonzero eigenvalues, and the vectors $v_1, \ldots, v_k$ will be the columns of Y, where v_i corresponds to the reciprocal of the eigenvalue to which u_i corresponds.
- Given the u_i's and v_i's, we can now solve a linear system to find the w_i's.

PARAFAC Decomposition

Tensor Decomposition

Input: tensor $T \in \mathbb{R}^{m \times n \times p}$ satisfying the conditions in Theorem $\boxed{3.1.3}$

Output: factors $\{u_i\}_i, \{v_i\}_i$ and $\{w_i\}_i$

Choose $a, b \in \mathbb{S}^{p-1}$ uniformly at random; set $T_a = T(*, *, a)$ and $T_b = T(*, *, b)$

Compute the eigendecomposition of $T_a(T_b)^+$ and $T_b(T_a)^+$

Let U and V be the eigenvectors

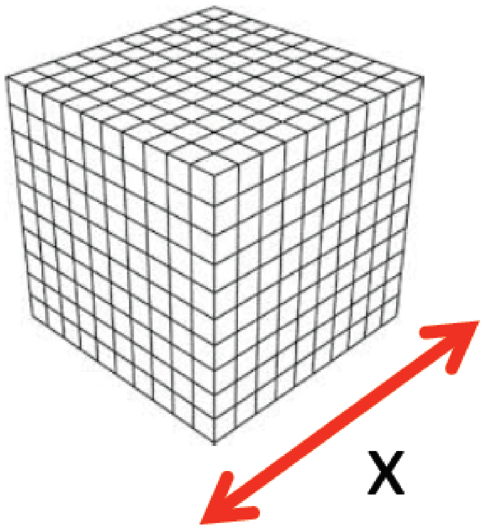
Pair up u_i and v_i iff their eigenvalues are reciprocals

Solve for w_i in $T = \sum_{i=1}^{r} u_i \otimes v_i \otimes w_i$

End

PARAFAC Decomposition

- Compute $T(\bullet, \bullet, x)$



i.e. add up matrix slices

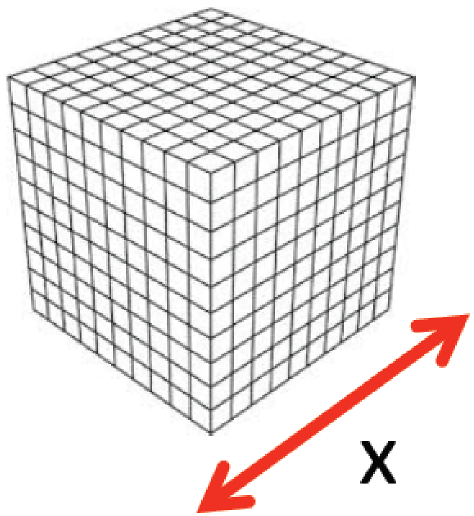
$$\sum x_i T_i$$

If $T = a \otimes b \otimes c$ then $T(\bullet, \bullet, x) = \langle c, x \rangle a \otimes b$

PARAFAC Decompo

$\mathrm{Diag}(\langle c_i, x \rangle)$

Compute $T(\bullet, \bullet, x) = A D_x B^T$



i.e. add up matrix slices

$$\sum x_i T_i$$

(x is chosen uniformly at random from S^{n-1})

PARAFAC Decomposition

- Compute $T(\bullet, \bullet, x) = A D_x B^T$
- Compute $T(\bullet, \bullet, y) = A D_y B^T$
- Diagonalize $\underbrace{T(\bullet, \bullet, x)\ T(\bullet, \bullet, y)^{-1}}_{A D_x B^T (B^T)^{-1} D_y^{-1} A^{-1}}$

PARAFAC Decomposition

- Compute $T(\bullet, \bullet, x) = A D_x B^T$
- Compute $T(\bullet, \bullet, y) = A D_y B^T$
- Diagonalize $\underbrace{T(\bullet, \bullet, x)\ T(\bullet, \bullet, y)^{-1}}_{A D_x D_y^{-1} A^{-1}}$

Claim: whp (over x,y) the eigenvalues are distinct, so the Eigendecomposition is unique and recovers a_i's

PARAFAC Decomposition

- Compute $T(\bullet, \bullet, x) = A D_x B^T$
- Compute $T(\bullet, \bullet, y) = A D_y B^T$
- Diagonalize $T(\bullet, \bullet, x)\ T(\bullet, \bullet, y)^{-1}$
- Diagonalize $T(\bullet, \bullet, y)\ T(\bullet, \bullet, x)^{-1}$
- Match up the factors (their eigenvalues are reciprocals) and find $\{c_i\}$ by solving a linear syst.

PARAFAC Decomposition

Given: $M = \sum a_i \otimes b_i$

When can we recover the factors a_i and b_i uniquely?

This is only possible if $\{a_i\}$ and $\{b_i\}$ are orthonormal, or rank(M)=1

Given: $T = \sum a_i \otimes b_i \otimes c_i$

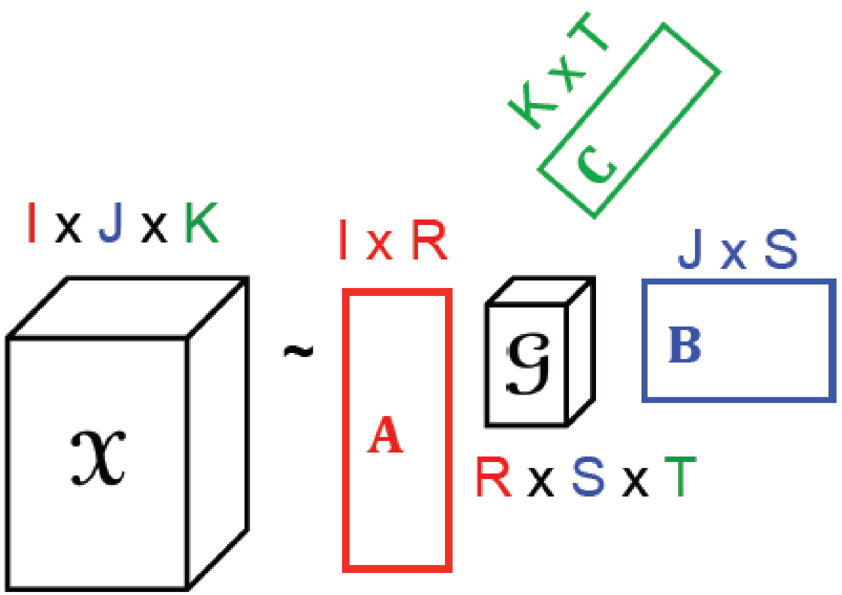
When can we recover the factors a_i, b_i and c_i uniquely?

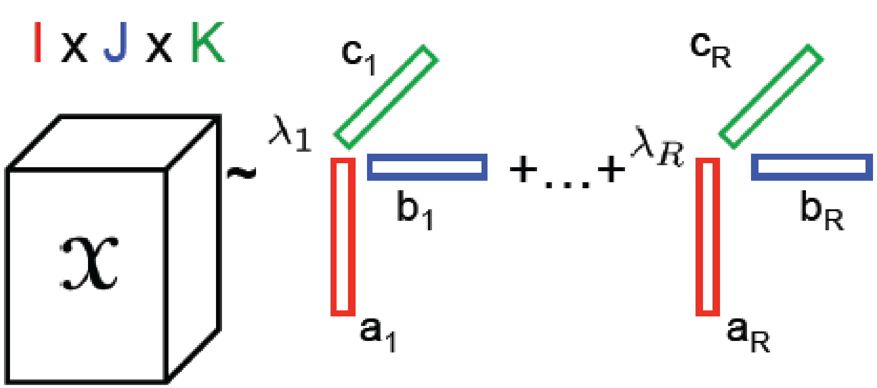
Jennrich: If $\{a_i\}$ and $\{b_i\}$ are full rank and no pair in $\{c_i\}$ are scalar multiples of each other

Tucker vs. PARAFAC Decomp.

- Tucker
  - Variable transformation in each mode
  - Core G may be dense
  - A, B, C generally orthonormal
  - Not unique

- PARAFAC
  - Sum of rank-1 components
  - No core, i.e., superdiagonal core
  - A, B, C may have linearly dependent columns
  - Generally unique





Tensor Decomposition Tools

- Two main tools
  - PARAFAC
  - Tucker
- Both find row-, column-, tube-groups
  - But in PARAFAC the three groups are identical
- To solve: Alternating Least Squares
- Toolbox: from Tamara Kolda:
  - http://csmr.ca.sandia.gov/~tgkolda/TensorToolbox/

Conclusions

- Real data are often in high dimensions with multiple aspects (modes)
- Matrices and tensors provide elegant theory and algorithms
- Tensor decompositions are unique under much more general conditions, compared to matrix decompositions
  - Tucker decomposition
  - Jennrich's algorithm (PARFAC, Kruskal tensors)
- Many applications

Reference:
Tensors https://web.stanford.edu/class/cs168/l/l10.pdf
Mining and Forecasting of Big Time-series Data, SIGMOD 2015 Tutorial
Algorithmic Aspects of Machine Learning, Ankur Moitra