

Compression of Higher-Dimensional Arrays using
the SVD

Householder Symposium 2005

Carla D. Moravitz Martin

(Joint work with Charles F. Van Loan)

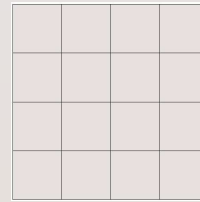
Center for Applied Mathematics

Cornell University

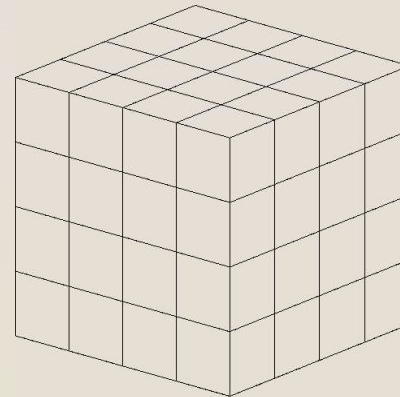
May 24, 2005

What are Tensors?

- Second-order tensor $A = (a_{ij}) \in \mathbb{R}^{n_1 \times n_2}$



- Third-order tensor $\mathcal{A} = (a_{ijk}) \in \mathbb{R}^{n_1 \times n_2 \times n_3}$



- p^{th} -order tensor $\mathcal{A} = (a_{i_1 i_2 \dots i_p}) \in \mathbb{R}^{n_1 \times \dots \times n_p}$

Leading Applications

- Chemometrics
- Psychometrics
- Computer Image Recognition
- Chromatography
- Other applications using multiway data analysis
(e.g., acoustics, phylogenetics)

Motivation: A two-way decomposition

Suppose $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ are orthogonal, and $\Sigma = U^T A V$, then

$$\begin{aligned} A = U \Sigma V^T &= \sum_{i=1}^m \sum_{j=1}^n \sigma_{ij} u_i v_j^T \\ &= \sum_{i=1}^m \sum_{j=1}^n \sigma_{ij} (u_i \circ v_j) \end{aligned}$$

where $u_i = U(:, i)$, $v_j = V(:, j)$

SVD Representation

Can choose $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$ orthogonal so
 $\Sigma = U^T A V = \text{diag}(\sigma_1, \dots, \sigma_r)$:

$$A = U \Sigma V^T = \sum_{i=1}^r \sigma_i u_i v_i^T = \sum_{i=1}^r \sigma_i (u_i \circ v_i)$$

where $u_i = U(:, i)$, $v_i = V(:, i)$ and $\text{rank}(A) = r$.

Reduction to diagonal form and rank-revealing properties make the SVD appealing to many applications.

Tensor Decompositions

Let $\mathcal{A} \in \mathbb{R}^{m \times n \times p}$

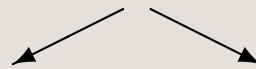
Goal: To find $U \in \mathbb{R}^{m \times m}$, $V \in \mathbb{R}^{n \times n}$, $W \in \mathbb{R}^{p \times p}$, and $\Sigma = (\sigma_{ijk}) \in \mathbb{R}^{m \times n \times p}$ such that

$$\mathcal{A} = \sum_{i=1}^m \sum_{j=1}^n \sum_{k=1}^p \sigma_{ijk} (u_i \circ v_j \circ w_k)$$

where $u_i = U(:, i)$, $v_j = V(:, j)$, $w_k = W(:, k)$

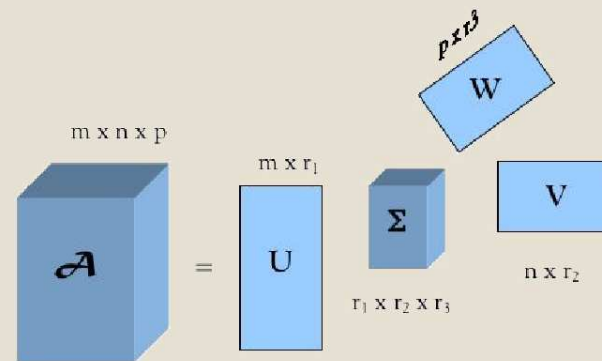
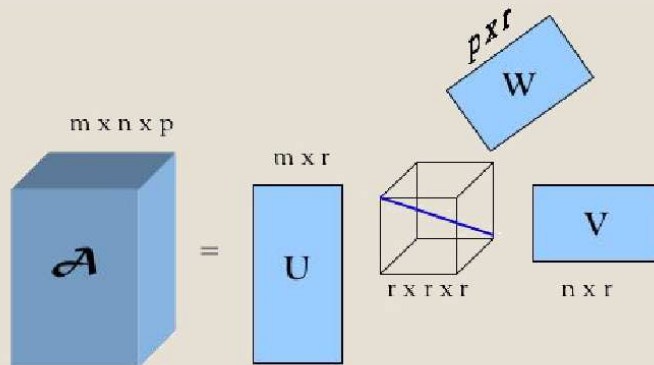
Orthogonal *or* Diagonal for Tensors

$$\begin{matrix} m \times n & m \times r & r \times r & r \times n \\ \mathbf{A} & = & \mathbf{U} & \mathbf{\Sigma} & \mathbf{V}^T \end{matrix}$$



Case 1: Diagonal Σ

Case 2: Orthogonal U, V, W



General Tensor Rank

Tensor rank of $\mathcal{A} \in \mathbb{R}^{m \times n \times p}$ is the minimum number of rank-1 tensors that sum to \mathcal{A} in linear combination.

If a tensor \mathcal{A} has a minimal representation as

$$\mathcal{A} = \sum_{i=1}^r \sigma_i (u_i \circ v_i \circ w_i),$$

then $\text{rank}(\mathcal{A}) = r$.

Tensor Rank is complicated ...

- Formula for the maximum possible rank of a tensor does not exist (as far as we know...)
- No known method to compute the “minimum” tensor decomposition
- Minimum tensor representation not necessarily orthogonal
- Set of rank-deficient tensors has positive volume

Why is Tensor Rank important?

- Enables data compression
- Identifies dependencies in data

Applications, multilinear algebra theory, and computational realities all have “something to say” about the tensor rank issue.

Goal of Compression

In the representation

$$\mathcal{A} = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sigma_{ijk} (u_i \circ v_j \circ w_k),$$

can we choose orthogonal $U, V, W \in \mathbb{R}^{n \times n}$ so that $\Sigma = (\sigma_{ijk})$ is “compressed”?

(i.e., Σ has most of its “mass” in a relatively small number of σ_{ijk})

Jacobi-Compression Algorithm

- Extension of Jacobi SVD Algorithm for matrices
- Computes tensor representations of $2 \times 2 \times 2$ subtensors

- Maximizes $\sum_{i=1}^n \sigma_{iii}^2$ or $\sum_{i=1}^n \sigma_{iii}$

Review of Jacobi SVD Algorithm

- Computes SVD of 2×2 submatrices of $A \in \mathbb{R}^{n \times n}$
- Pick a (p, q) pair and compute (c_1, s_1) , (c_2, s_2) so that

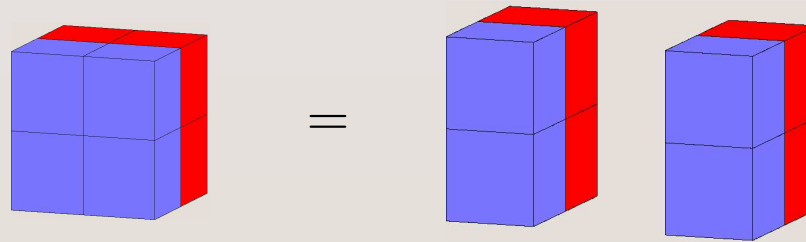
$$\begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix}^T \begin{bmatrix} a_{pp} & a_{pq} \\ a_{qp} & a_{qq} \end{bmatrix} \begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix} = \begin{bmatrix} \sigma_p & 0 \\ 0 & \sigma_q \end{bmatrix}$$

- Update affected portions of A

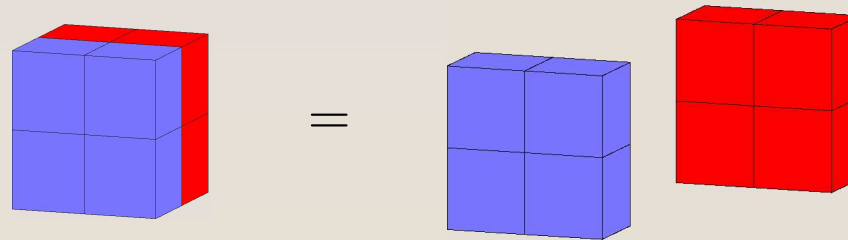
Representation of $2 \times 2 \times 2$ Tensors

Three ways to cut a “cube”:

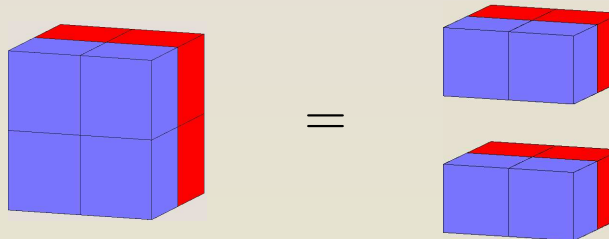
1. Left-right



2. Front-back



3. Top-bottom



Another $2 \times 2 \times 2$ Representation

Representing tensors as vectors:

The diagram illustrates the process of representing a 2x2x2 tensor \mathcal{A} as a vector. On the left, a 3D cube is shown with a blue front face and a red back face. This is equated to two 2x2 matrices: a blue matrix with elements a_{111} , a_{121} , a_{211} , and a_{221} , and a red matrix with elements a_{112} , a_{122} , a_{212} , and a_{222} . To the right, the vector $\text{vec}(\mathcal{A})$ is shown as a column vector containing the elements of the blue matrix followed by the elements of the red matrix: a_{111} , a_{211} , a_{121} , a_{221} , a_{112} , a_{212} , a_{122} , and a_{222} .

$$\mathcal{A} = \begin{bmatrix} a_{111} & a_{121} \\ a_{211} & a_{221} \end{bmatrix} \begin{bmatrix} a_{112} & a_{122} \\ a_{212} & a_{222} \end{bmatrix}$$
$$\text{vec}(\mathcal{A}) = \begin{bmatrix} a_{111} \\ a_{211} \\ a_{121} \\ a_{221} \\ a_{112} \\ a_{212} \\ a_{122} \\ a_{222} \end{bmatrix}$$

where vec stacks the columns of the faces of the tensor.

Tensor Decompositions as matrix-vector products

Note,

$$\mathcal{A} = \sum_{i=1}^n \sum_{j=1}^n \sum_{k=1}^n \sigma_{ijk} (u_i \circ v_j \circ w_k),$$

$$\Updownarrow$$

$$a = (W \otimes V \otimes U) \cdot \sigma$$

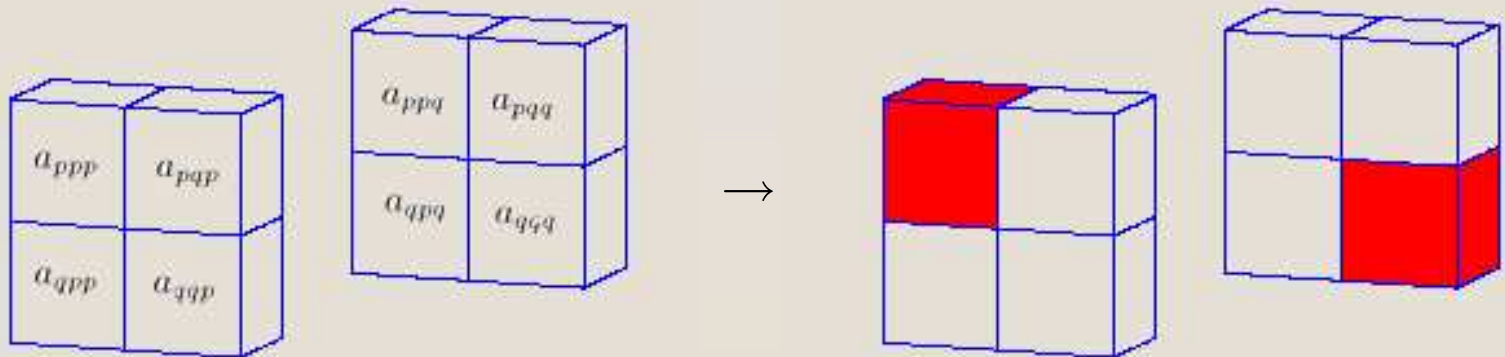
$$\Updownarrow$$

$$\sigma = (W^T \otimes V^T \otimes U^T) \cdot a$$

where $a = \text{vec}(\mathcal{A})$ and $\sigma = \text{vec}(\sigma_{ijk})$

Jacobi-Compression Algorithm

Works with (p, q) pairs of $2 \times 2 \times 2$ subtensors of $\mathcal{A} \in \mathbb{R}^{n \times n \times n}$:



Faster convergence if a *sweep* involves three different cube orientations

Solving the Subproblem

The objective function can be written out explicitly using

$$U = \begin{bmatrix} c_1 & s_1 \\ -s_1 & c_1 \end{bmatrix} \quad V = \begin{bmatrix} c_2 & s_2 \\ -s_2 & c_2 \end{bmatrix} \quad W = \begin{bmatrix} c_3 & s_3 \\ -s_3 & c_3 \end{bmatrix}$$

- Alternating Least Squares idea to maximize $\sum_{i=1}^n \sigma_{iii}^2$ or $\sum_{i=1}^n \sigma_{iii}$

Solving the Subproblem

Alternating Least Squares idea:

$$\text{Step 1.} \quad \sigma_1 \leftarrow (I \otimes V_1^T \otimes U_1^T) \cdot a$$

$$\text{Step 2.} \quad \sigma_2 \leftarrow (W_1^T \otimes I \otimes U_2^T) \cdot \sigma_1$$

$$\text{Step 3.} \quad \sigma_3 \leftarrow (W_2^T \otimes V_2^T \otimes I) \cdot \sigma_2$$

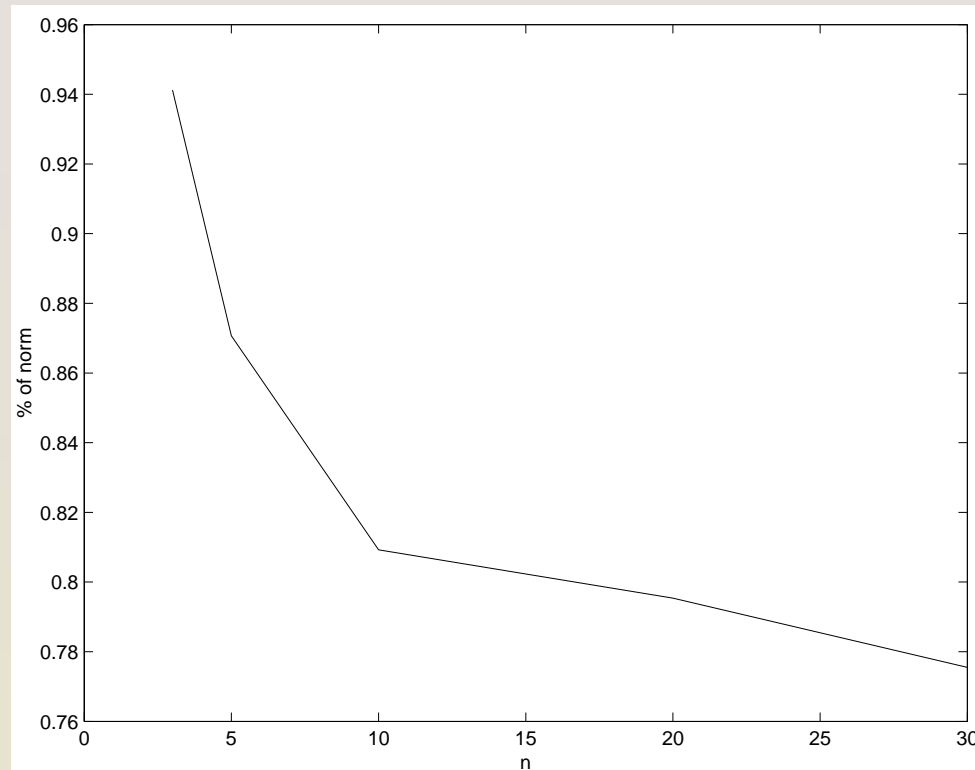
where “diagonals” are maximized at each step.

Why Alternating Least Squares?

- Maximizing $\sum_{i=1}^2 \sigma_{iii}^2$ has an explicit (yet complicated) solution
- Maximizing $\sum_{i=1}^2 \sigma_{iii}$ has an explicit SVD solution
- Easier to extend to higher dimensions

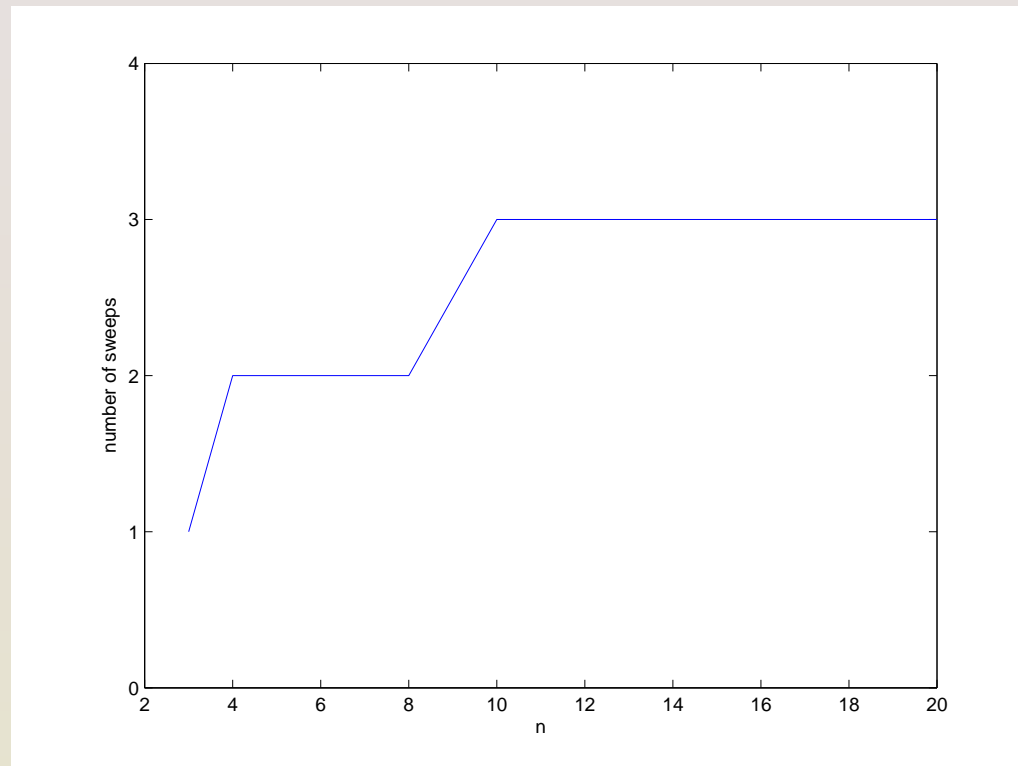
Diagonal Elements from Jacobi-Compression Algorithm

Compression for different n : $\left(\frac{\sum \sigma_{iii}^2}{\sum \sigma_{ijk}^2} \right)$



Convergence Results

Number of sweeps for convergence for orthogonally diagonalizable tensors for different n



Jacobi-Compression Algorithm

- Can be extended to $m \times n \times p$ -case by padding with zeros
- Can be implemented in parallel

Conclusions

- Applications need to drive the algorithms
- Can bypass the rank problem in some applications by “compressing” tensors
- Compression algorithm based on Jacobi-SVD algorithm and has potential to be implemented in parallel for large-scale problems