Tensor Computations and Applications in Data Mining

Lars Eldén

Department of Mathematics
Linköping University, Sweden
Joint work with Berkant Savas

SIAM AM July 2008

Are Tensors too Difficult?

Murray & Rice, Differential geometry and statistics, 1993:

$$\xi(\chi)^{i_1 \ldots i_r}_{j_1 \ldots j_s} = \xi(\theta)^{k_1 \ldots k_r}_{l_1 \ldots l_s} \frac{\partial \chi^{i_1}}{\partial \theta^{k_1}} \cdots \frac{\partial \chi^{i_r}}{\partial \theta^{k_r}} \frac{\partial \theta^{l_1}}{\partial \chi^{k_1}} \cdots \frac{\partial \theta^{j_1}}{\partial \chi^{j_s}} \qquad (8.7.1)$$

Classically it would have been said that the tensor transforms by this rule. It is horrible formulae like this that have given tensor analysis a bad name.

"... the manipulation of matrices is a hundred times better supported in our brains and in our software tools than that of tensors."
(N. Trefethen, Maxims about numerical mathematics, science, computers, and life on earth)

Notation and Concepts

We need a notational and conceptual framework that

- exhibits the structure of the problems
- is independent of the order of the tensor, or easily generalizable
- allows the formulation and implementation of algorithms

Q: Can we find such a framework in math books on tensor calculus?

Notation and Concepts

We need a notational and conceptual framework that

- exhibits the structure of the problems
- is independent of the order of the tensor, or easily generalizable
- allows the formulation and implementation of algorithms

Q: Can we find such a framework in math books on tensor calculus?
A: NO! (in general), because we are asking different questions now.
Many fundamental mathematical problems are open!

Notation and Concepts

We need a notational and conceptual framework that

- exhibits the structure of the problems
- is independent of the order of the tensor, or easily generalizable
- allows the formulation and implementation of algorithms

Q: Can we find such a framework in math books on tensor calculus?
A: NO! (in general), because we are asking different questions now.
Many fundamental mathematical problems are open!

Tensor methods have been used since the 1960's in psychometrics and chemometrics! Only recently in numerical community.
Applications in signal processing and various areas of data mining.

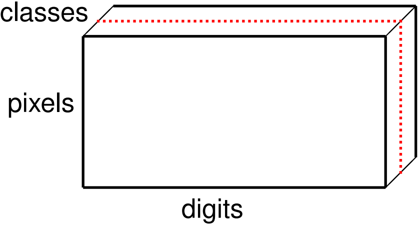
Recent survey:
Tammy Kolda & Brett Bader, Tensor Decompositions and Applications, SIAM Review, to appear. (Download from Tammy's web page)

Outline

Multi-Mode Data: Tensors

Example: Classification of hand-written digits

3–tensor $\mathcal{D}$ with
- pixel mode, 400 pixels
- digit mode, $\sim$ 1000 digits per class
- class mode, 10 classes

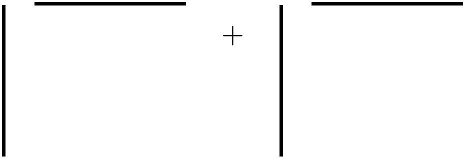


All digits of one class represented by a slice

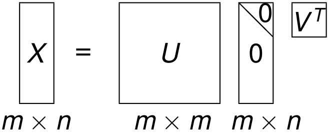
Two Aspects of SVD: Expansion – Decomposition

1. Expansion in terms of rank-1 matrices:

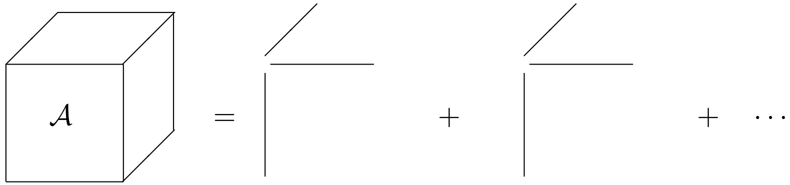
$$X = \sum_{i=1}^{n} \sigma_i u_i v_i^T = \qquad + \qquad + \quad \cdots$$



2. Matrix decomposition: $\mathbb{R}^{m \times n} \ni X = U \Sigma V^T$

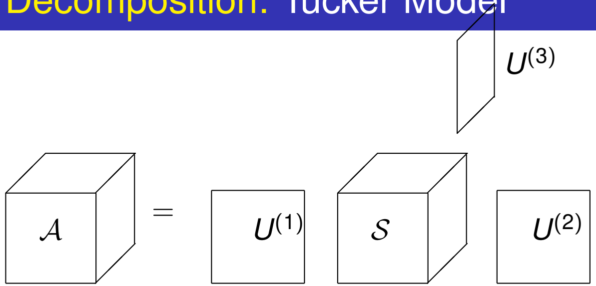


Tensor Expansion in Rank-1 Terms



- Parafac/Candecomp/Kruskal: Harshman, Caroll, Chang 1970
- Numerous papers in psychometrics and chemometrics
- From a mathematical point of view: difficult problem, sometimes ill-posed, see De Silva and Lim 2006.
- From the point of view of applications: very useful! (Rasmus Bro's talk)

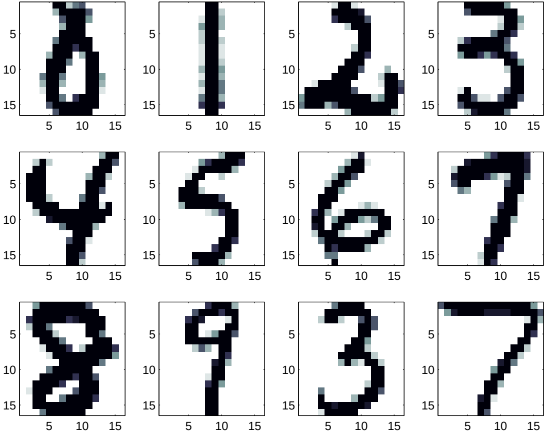
Tensor Decomposition: Tucker Model



- Tucker 1964, numerous papers in psychometrics and chemometrics
- De Lathauwer, De Moor, Vandewalle, SIMAX 2000: notation, theory.
- The matrices $U^{(i)}$ are usually orthogonal.

This talk: Tucker model for 3-tensors only!

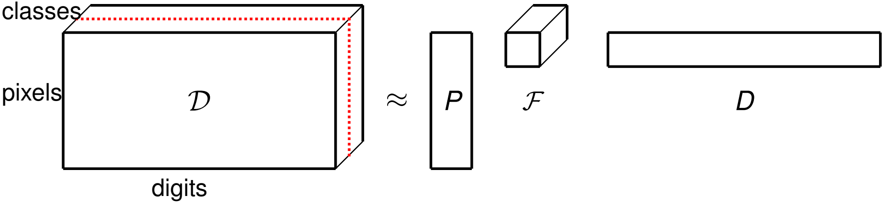
Classification of Handwritten Digits



"Model problem" in pattern recognition

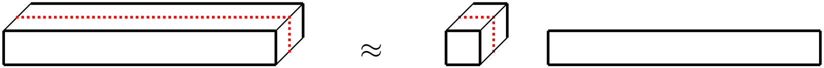
HOSVD for Data Reduction

pixel mode, 400 pixels
digit mode, $\sim$ 1000 digits per class
class mode, 10 classes



Cf. low-rank approximation of matrix by SVD: $A \approx U_k \Sigma_k V_k^T$

Project all Digits to Low Dimension



$(P^T)_1 \cdot \mathcal{D}$ $\approx$ $\mathcal{F}$ D

Each column is a digit in low dimension

10 class bases

Coordinates

Slice μ of $\mathcal{F}$ is a basis for class μ

Compute the SVD of each slice: $\mathcal{F}(:,:,\mu) = U^{\mu}\Sigma^{\mu}(V^{\mu})^T$ and use k columns, U_k^{μ}, as basis vectors.

Classification with HOSVD Compression

- Training phase:
  1. Collect the training digits into a tensor $\mathcal{D}$.
  2. Compute the HOSVD of $\mathcal{D}$.
  3. Compute the low rank “basis” tensor $\mathcal{F} = (P^T)_1 \cdot \mathcal{D}$.
  4. Compute and store the basis matrices $B^\mu = U_k^\mu$ for each class.
- Test phase: For each test digit d
  1. Project $d = P^T d$.
  2. Compute the residuals $R(\mu) = \|(I - B^\mu (B^\mu)^T) d\|$, $\mu = 1, \ldots, 10$.
  3. Determine $\mu_{\min} = \operatorname{argmin}_\mu R(\mu)$ and classify d as $\mu_{\min}$.

Classification results: US Postal Service Database

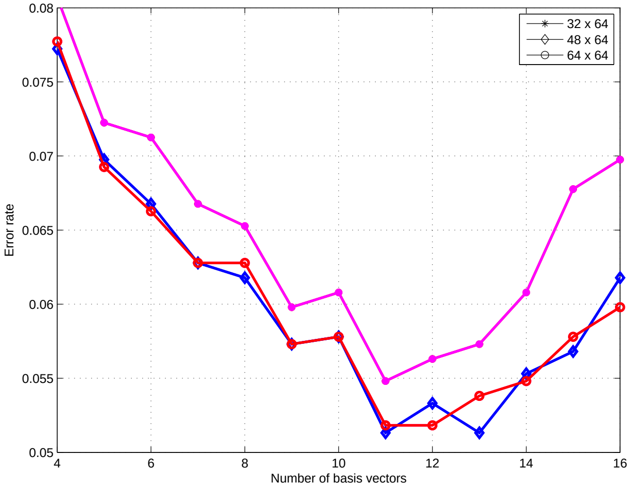


Figure: Error rates for different compressions ($> 97.8\%$), and basis dimension.

Mode$-I$ Multiplication of a Tensor by a Matrix

Assume that dimensions are such that all operations are well-defined.
Mostly 3-tensors. Lim's notation. (No standard notation yet)

$$\mathcal{B} = (X)_1 \cdot \mathcal{A}, \qquad \mathcal{B}(i,j,k) = \sum_{\nu=1}^{n} x_{i\nu} a_{\nu jk}.$$

All column vectors are multiplied by the matrix X.
Multiplication in all modes at the same time:

$$\mathcal{B} = (X, Y, Z) \cdot \mathcal{A}, \qquad \mathcal{B}(i,j,k) = \sum_{\nu,\mu,\lambda} x_{i\nu} y_{j\mu} z_{k\lambda} a_{\nu\mu\lambda}.$$

For convenience we write

$$\mathcal{B} = (X^T, Y^T, Z^T) \cdot \mathcal{A} = \mathcal{A} \cdot (X, Y, Z)$$

Inner Product and Norm

Inner product (contraction: $\mathbb{R}^{n \times n \times n} \to \mathbb{R}$)

$$\langle \mathcal{A}, \mathcal{B} \rangle = \sum_{i,j,k} a_{ijk} b_{ijk}$$

The Frobenius norm:

$$\|\mathcal{A}\| = \langle \mathcal{A}, \mathcal{A} \rangle^{1/2}$$

Matrix case

$$\langle A, B \rangle = \mathrm{tr}(A^T B)$$

Partial Contractions

$$\mathcal{C} = \langle \mathcal{A}, \mathcal{B} \rangle_1 \ , \qquad c_{jklm} = \sum_{\lambda} a_{\lambda jk} b_{\lambda lm} \, , \qquad \text{(4-tensor)} \, ,$$

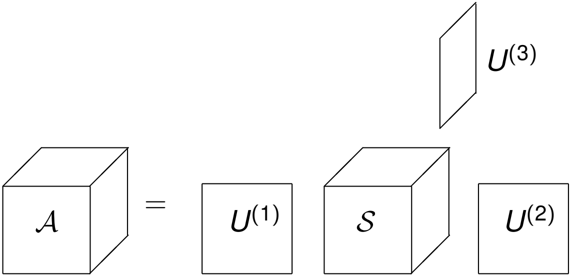
$$D = \langle \mathcal{A}, \mathcal{B} \rangle_{1:2} \ , \qquad d_{jk} = \sum_{\lambda,\mu} a_{\lambda\mu j} b_{\lambda\mu k} \, , \qquad \text{(2-tensor)},$$

$$e = \langle \mathcal{A}, \mathcal{B} \rangle = \langle \mathcal{A}, \mathcal{B} \rangle_{1:3} \ , \qquad e = \sum_{\lambda,\mu,\nu} a_{\lambda\mu\nu} b_{\lambda\mu\nu} \, , \qquad \text{(scalar)}.$$

Notation (3-tensor):

$$\langle \mathcal{A}, \mathcal{B} \rangle_{1:2} = \langle \mathcal{A}, \mathcal{B} \rangle_{-3}$$

Tensor SVD (HOSVD): $\mathcal{A} = (U^{(1)}, U^{(2)}, U^{(3)}) \cdot \mathcal{S}$

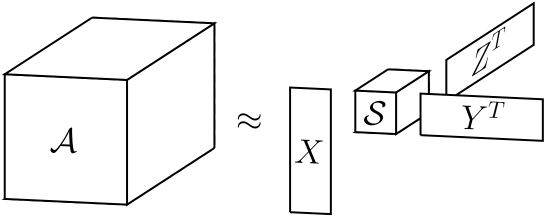


1. Compute the SVD of all mode$-i$ vectors
2. $U^{(i)}$ is left singular matrix of mode i
3. $\mathcal{S} := \mathcal{A} \cdot \left(U^{(1)}, U^{(2)}, U^{(3)}\right)$

The "mass" of $\mathcal{S}$ is concentrated around the $(1, 1, 1)$ corner.
Not optimal: does not give the solution of $\min_{\mathrm{rank}(\mathcal{B})=(r_1,r_2,r_3)} \|\mathcal{A} - \mathcal{B}\|$

De Lathauwer et al (2000)

Best Rank$-(r_1, r_2, r_3)$ Approximation



Best rank$-(r_1, r_2, r_3)$ approximation:

$$\min_{X,Y,Z,\mathcal{S}} \|\mathcal{A} - (X, Y, Z) \cdot \mathcal{S}\|, \qquad X^T X = I, \quad Y^T Y = I, \quad Z^T Z = I$$

The problem is over-parameterized!

Best Approximation

$$\min_{\text{rank}(\mathcal{B})=(r_1,r_2,r_3)} \|\mathcal{A} - \mathcal{B}\|$$

is equivalent to

$$\max_{X,Y,Z} \Phi(X,Y,Z) = \frac{1}{2}\|\mathcal{A}\cdot(X,Y,Z)\|^2$$

$$= \frac{1}{2}\sum_{j,k,l}\left(\sum_{\lambda,\mu,\nu} a_{\lambda\mu\nu}x_{\lambda j}y_{\mu k}z_{\nu l}\right)^2,$$

subject to

$$X^TX = I_{r_1}, \qquad Y^TY = I_{r_2}, \qquad Z^TZ = I_{r_3}$$

Grassmann Optimization

The Frobenius norm is invariant under orthogonal transformations:

$$\Phi(X, Y, Z) = \Phi(XU, YV, ZW) = \frac{1}{2}\|\mathcal{A} \cdot (XU, YV, ZW)\|^2$$

for orthogonal $U \in \mathbb{R}^{r_1 \times r_1}$, $V \in \mathbb{R}^{r_2 \times r_2}$, and $W \in \mathbb{R}^{r_3 \times r_3}$.
Maximize Φ over equivalence classes

$$[X] = \{XU \mid U \text{ orthogonal}\}.$$

Product of manifolds: $\mathrm{Gr}^3 = \mathrm{Gr}(J, r_1) \times \mathrm{Gr}(K, r_2) \times \mathrm{Gr}(L, r_3)$

$$\max_{(X,Y,Z)\in\mathrm{Gr}^3} \Phi(X, Y, Z) = \max_{(X,Y,Z)\in\mathrm{Gr}^3} \frac{1}{2}\langle \mathcal{A} \cdot (X, Y, Z), \mathcal{A} \cdot (X, Y, Z)\rangle$$

Newton's Method on one Grassmann Manifold

Taylor expansion + linear algebra on tangent space[1] at X

$$G(X(t)) \approx G(X(0)) + \langle \Delta, \nabla G \rangle + \frac{1}{2}\langle \Delta, H(\Delta) \rangle,$$

Grassmann gradient:

$$\nabla G = \Pi_X G_x, \qquad (G_x)_{jk} = \frac{\partial G}{\partial x_{jk}}, \qquad \Pi_X = I - XX^T$$

The Newton equation for determining Δ:

$$\Pi_X \langle \mathcal{G}_{xx}, \Delta \rangle_{1:2} - \Delta \langle X, G_x \rangle_1 = -\nabla G, \qquad (\mathcal{G}_{xx})_{jklm} = \frac{\partial^2 G}{\partial X_{jk}\, \partial X_{lm}}.$$

[1] Tangent space at X: all matrices Z satisfying $Z^T X = 0$.

Newton-Grassmann Algorithm on Gr^3

Here: local coordinates

Given tensor $\mathcal{A}$ and starting points $(X_0, Y_0, Z_0) \in \mathrm{Gr}^3$

repeat

1. compute the Grassmann gradient $\widehat{\nabla\Phi}$
2. compute the Grassmann Hessian $\widehat{\mathcal{H}}$
3. matricize $\widehat{\mathcal{H}}$ and vectorize $\widehat{\nabla\Phi}$
4. solve $D = (D_x, D_y, D_z)$ from the Newton equation
5. take a geodesic step along the direction D, giving new iterates (X,Y,Z)

until $\|\widehat{\nabla\Phi}\|/\Phi <$ TOL

Implementation using TensorToolbox (Bader/Kolda) and home-made object-oriented Grassmann classes in Matlab

Newton's method on Gr^3

Differentiate $\Phi(X, Y, Z)$ along a geodesic curve $(X(t), Y(t), Z(t))$ in the direction $(\Delta_x, \Delta_y, \Delta_z)$:

$$\frac{\partial x_{st}}{\partial t} = (\Delta_x)_{st},$$

and

$$\left(\frac{dX(t)}{dt}, \frac{dY(t)}{dt}, \frac{dZ(t)}{dt}\right) = (\Delta_x\,, \Delta_y\,, \Delta_z),$$

Since $\mathcal{A} \cdot (X, Y, Z)$ is linear in X, Y, Z separately:

$$\frac{d(\mathcal{A} \cdot (X, Y, Z))}{dt} = \mathcal{A} \cdot (\Delta_x, Y, Z) + \mathcal{A} \cdot (X, \Delta_y, Z) + \mathcal{A} \cdot (X, Y, \Delta_z).$$

First Derivative

$$\frac{d\Phi}{dt} = \frac{1}{2}\frac{d}{dt}\langle \mathcal{A}\cdot(X,Y,Z), \mathcal{A}\cdot(X,Y,Z)\rangle = \langle \mathcal{A}\cdot(\Delta_x,Y,Z), \mathcal{A}\cdot(X,Y,Z)\rangle$$
$$+\langle \mathcal{A}\cdot(X,\Delta_y,Z), \mathcal{A}\cdot(X,Y,Z)\rangle + \langle \mathcal{A}\cdot(X,Y,\Delta_z), \mathcal{A}\cdot(X,Y,Z)\rangle.$$

We want to write $\langle \mathcal{A}\cdot(\Delta_x,Y,Z), \mathcal{A}\cdot(X,Y,Z)\rangle$ in the form $\langle \Delta_x, \Phi_x\rangle$
Define the tensor $\mathcal{F} = \mathcal{A}\cdot(X,Y,Z)$ and write

$$\langle \mathcal{A}\cdot(\Delta_x,Y,Z), \mathcal{F}\rangle =: \langle \mathcal{K}_x(\Delta_x), \mathcal{F}\rangle = \langle \Delta_x, \mathcal{K}_x^*\mathcal{F}\rangle,$$

For fixed Y and Z we have a linear operator:

$$\Delta_x \longmapsto \mathcal{K}_x(\Delta_x) = \mathcal{A}\cdot(\Delta_x,Y,Z)$$

Adjoint Operator

Linear operator:

$$\Delta_x \longmapsto \mathcal{K}_x(\Delta_x) = \mathcal{A} \cdot (\Delta_x, Y, Z)$$

with adjoint

$$\langle \mathcal{K}_x(\Delta_x), \mathcal{F} \rangle = \langle \Delta_x, \mathcal{K}_x^* \mathcal{F} \rangle = \langle \Delta_x, \langle \mathcal{A} \cdot (I, Y, Z), \mathcal{F} \rangle_{-1} \rangle$$

where the partial contraction is defined

$$\langle \mathcal{B}, \mathcal{C} \rangle_{-1}(i_1, i_2) = \sum_{\mu,\nu} b_{i_1 \mu\nu} c_{i_2 \mu\nu}$$

Grassmann Gradient

X-part: multiply by $\Pi_x = I - XX^T$

$$\begin{aligned}\Pi_X \Phi_x &= \Pi_X \langle \mathcal{A} \cdot (I, Y, Z), \mathcal{F} \rangle_{-1} \\ &= \langle \mathcal{A} \cdot (I, Y, Z), \mathcal{A} \cdot (X, Y, Z) \rangle_{-1} - XX^T \langle \mathcal{A} \cdot (I, Y, Z), \mathcal{F} \rangle_{-1} \\ &= \langle \mathcal{A} \cdot (I, Y, Z), \mathcal{A} \cdot (I, Y, Z) \rangle_{-1} X - X \langle \mathcal{F}, \mathcal{F} \rangle_{-1},\end{aligned}$$

Complete gradient (recall $\mathcal{F} = \mathcal{A} \cdot (X, Y, Z)$):

$$\nabla \Phi = (\Pi_X \Phi_x, \Pi_Y \Phi_y, \Pi_Z \Phi_z),$$

where

$$\Pi_X \Phi_x = \langle \mathcal{A} \cdot (I, Y, Z), \mathcal{A} \cdot (I, Y, Z) \rangle_{-1} X - X \langle \mathcal{F}, \mathcal{F} \rangle_{-1}$$

$$\Pi_Y \Phi_y = \langle \mathcal{A} \cdot (X, I, Z), \mathcal{A} \cdot (X, I, Z) \rangle_{-2} Y - Y \langle \mathcal{F}, \mathcal{F} \rangle_{-2}$$

$$\Pi_Y \Phi_z = \langle \mathcal{A} \cdot (X, Y, I), \mathcal{A} \cdot (X, Y, I) \rangle_{-3} Z - Z \langle \mathcal{F}, \mathcal{F} \rangle_{-3}$$

Second Derivative

$$
\begin{aligned}
&\frac{d^2\Phi}{dt^2} = \\
&= \langle \mathcal{A} \cdot (\Delta_x, Y, Z), \mathcal{A} \cdot (\Delta_x, Y, Z) \rangle + \langle \mathcal{A} \cdot (\Delta_x, \Delta_y, Z), \mathcal{A} \cdot (X, Y, Z) \rangle \\
&+ \langle \mathcal{A} \cdot (\Delta_x, Y, Z), \mathcal{A} \cdot (X, \Delta_y, Z) \rangle + \langle \mathcal{A} \cdot (\Delta_x, Y, \Delta_z), \mathcal{A} \cdot (X, Y, Z) \rangle \\
&+ \langle \mathcal{A} \cdot (\Delta_x, Y, Z), \mathcal{A} \cdot (X, Y, \Delta_z) \rangle + \cdots ,
\end{aligned}
$$

plus 10 analogous terms.

Grassmann Hessian

$$\mathcal{H}(\Delta) = (\Phi_{x*}(\Delta), \Phi_{y*}(\Delta), \Phi_{z*}(\Delta)) \; : \; \mathbb{T}^3 \mapsto \mathbb{T}^3,$$

where

$$\begin{aligned}
\Phi_{x*}(\Delta) &= \mathcal{H}_{xx}(\Delta_x) + \mathcal{H}_{xy}(\Delta_y) + \mathcal{H}_{xz}(\Delta_z), \qquad & \Phi_{x*}(\cdot) \; &: \; \mathbb{T}^3 \to \mathbb{T}_X, \\
\Phi_{y*}(\Delta) &= \mathcal{H}_{yx}(\Delta_x) + \mathcal{H}_{yy}(\Delta_y) + \mathcal{H}_{yz}(\Delta_z), \qquad & \Phi_{y*}(\cdot) \; &: \; \mathbb{T}^3 \to \mathbb{T}_Y, \\
\Phi_{z*}(\Delta) &= \mathcal{H}_{zx}(\Delta_x) + \mathcal{H}_{zy}(\Delta_y) + \mathcal{H}_{zz}(\Delta_z), \qquad & \Phi_{z*}(\cdot) \; &: \; \mathbb{T}^3 \to \mathbb{T}_Z,
\end{aligned}$$

Grassmann Hessian, "Diagonal Part"

$$
\begin{aligned}
\mathcal{H}_{xx}(\Delta_x) &= \Pi_X \langle \mathcal{B}_x, \mathcal{B}_x \rangle_{-1} \Delta_x - \Delta_x \langle \mathcal{F}, \mathcal{F} \rangle_{-1}, \quad & \mathcal{B}_x &= \mathcal{A} \cdot (I, Y, Z), \\
\mathcal{H}_{yy}(\Delta_y) &= \Pi_Y \langle \mathcal{B}_y, \mathcal{B}_y \rangle_{-2} \Delta_y - \Delta_y \langle \mathcal{F}, \mathcal{F} \rangle_{-2}, \quad & \mathcal{B}_y &= \mathcal{A} \cdot (X, I, Z), \\
\mathcal{H}_{zz}(\Delta_z) &= \Pi_Z \langle \mathcal{B}_z, \mathcal{B}_z \rangle_{-3} \Delta_z - \Delta_z \langle \mathcal{F}, \mathcal{F} \rangle_{-3}, \quad & \mathcal{B}_z &= \mathcal{A} \cdot (X, Y, I).
\end{aligned}
$$

Grassmann Hessian, "Upper Triangular Part",

$$\mathcal{H}_{xy}(\Delta_y) = \Pi_X \left(\left\langle \langle \mathcal{C}_{xy}, \mathcal{F} \rangle_{-(1,2)} , \Delta_y \right\rangle_{2,4;1,2} + \left\langle \langle \mathcal{B}_x, \mathcal{B}_y \rangle_{-(1,2)} , \Delta_y \right\rangle_{4,2;1,2} \right),$$

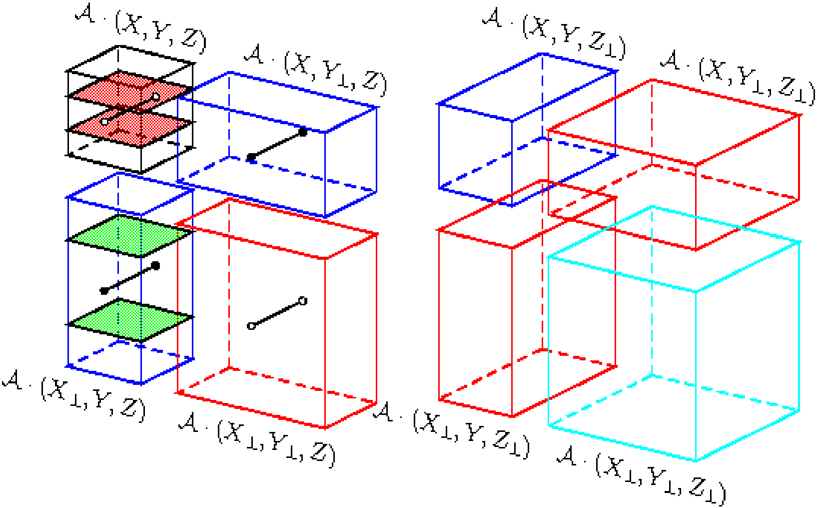
where $\mathcal{C}_{xy} = \mathcal{A} \cdot (I, I, Z)$, etc.

4-tensor contracted with a matrix giving a matrix:

$$\left\langle \langle \mathcal{C}_{xy}, \mathcal{F} \rangle_{-(1,2)} , \Delta_y \right\rangle_{2,4;1,2}$$

Illustration of Hessian Computation

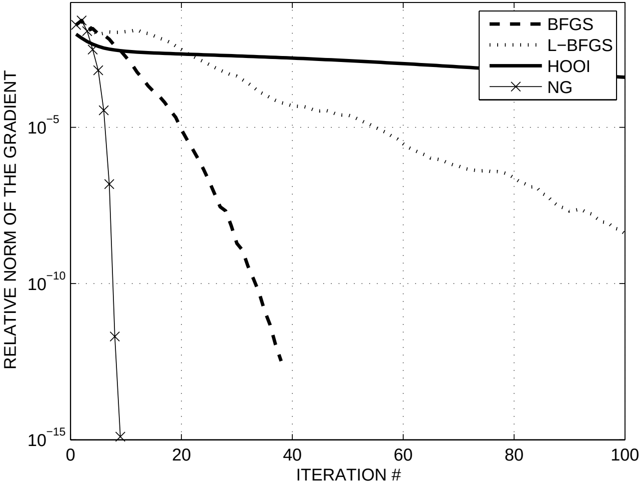
Local coordinates.



Methods for Best Approximation

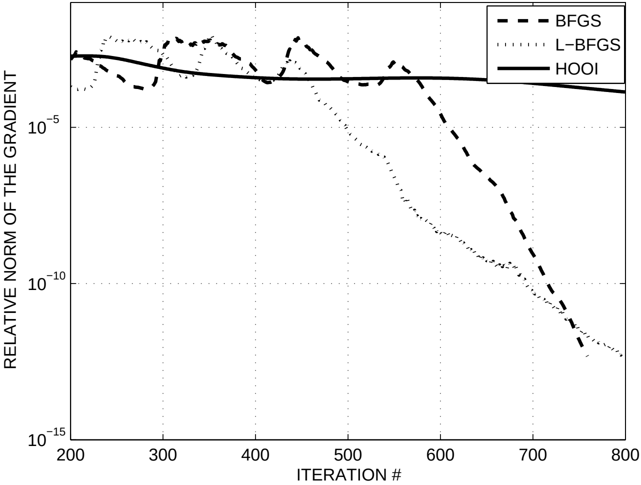
- Grassmann-based
  1. Newton (LE, B. Savas)
  2. Trust region/Newton (Ishteva, De Lathauwer et al.)
  3. BFGS quasi-Newton (Savas, Lim)
  4. Limited memory BFGS (Savas, Lim)
- Alternating
  1. HOOI (Kroonenberg, De Lathauwer)

Numerical Example I



A random tensor $\mathcal{A} \in \mathbb{R}^{20\times20\times20}$ with random entries N(0, 1) approximated with a rank $-(5, 5, 5)$ tensor.

Numerical Example II



A random tensor $\mathcal{A} \in \mathbb{R}^{100\times100\times100}$ with random entries N(0, 1) approximated with a rank $-(5, 10, 20)$ tensor.

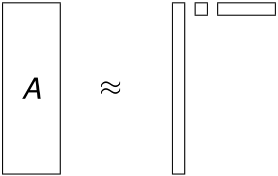
Sparse Tensors in Information Sciences

In information sciences the tensors are often sparse:

- Term-document-author (Dunlavy et al)
- Graphs, web link analysis (Kolda et al)

For sparse matrices: Krylov methods give low rank approximations:

$$AV_k = U_k H_k$$

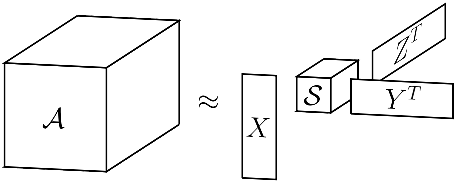


$$= U_k H_k V_k^T.$$

The matrix is only used as operator: $u = Av$

Sparse Tensors

Can we generalize Krylov methods to tensors and obtain low rank approximations?



Golub-Kahan Bidiagonalization for Rectangular Matrix

- $\beta_1 u_1 = b$, $v_0 = 0$
- **for** $i = 1 : k$

$$\alpha_i v_i = A^T u_i - \beta_i v_{i-1},$$
$$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i$$

- **end**

The coefficients α_i and β_i are chosen to normalize the vectors.

Golub-Kahan Bidiagonalization for Rectangular Matrix

- $\beta_1 u_1 = b$, $v_0 = 0$
- **for** $i = 1 : k$

$$\alpha_i v_i = A^T u_i - \beta_i v_{i-1}, \qquad [\alpha_i v_i = A \cdot (u_i)_1 - \beta_i v_{i-1},]$$

$$\beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i \qquad [\beta_{i+1} u_{i+1} = A \cdot (v_i)_2 - \alpha_i u_i]$$

- **end**

The coefficients α_i and β_i are chosen to normalize the vectors.

Krylov Method for Tensor Approximation

Arnoldi style (i.e., including Gram-Schmidt orthogonalization)

- Let u_1 and v_1 be given
- $h_{111} w_1 = \mathcal{A} \cdot (u_1, v_1)_{1,2}$
- **for** $\nu = 2 : m$

$$
\begin{aligned}
&h_u = \mathcal{A} \cdot (U_{\nu-1}, v_{\nu-1}, w_{\nu-1}) \\
&h_{\nu,\nu-1,\nu-1} u_\nu = \mathcal{A} \cdot (v_{\nu-1}, w_{\nu-1})_{2,3} - U_{\nu-1} h_u \\
&h_v = \mathcal{A} \cdot (u_\nu, V_{\nu-1}, w_{\nu-1}) \\
&h_{\nu,\nu,\nu-1} v_\nu = \mathcal{A} \cdot (u_\nu, w_{\nu-1})_{1,3} - V_{\nu-1} h_v \\
&h_w = \mathcal{A} \cdot (u_\nu, v_\nu, W_{\nu-1}) \\
&h_{\nu\nu\nu} w_\nu = \mathcal{A} \cdot (u_\nu, v_\nu)_{1,2} - W_{\nu-1} h_w
\end{aligned}
$$

- **end**

Approximate

$$
\mathcal{A} \approx (U_m, V_m, W_m) \cdot \mathcal{H}, \qquad \mathcal{H} = \left(U_m^T, V_m^T, W_m^T \right) \cdot \mathcal{A}
$$

Krylov Method for Tensor Approximation

Arnoldi style (i.e., including Gram-Schmidt orthogonalization)

- Let u_1 and v_1 be given
- $h_{111} w_1 = \mathcal{A} \cdot (u_1, v_1)_{1,2}$
- **for** $\nu = 2 : m$

$$
\begin{aligned}
&h_u = \mathcal{A} \cdot (U_{\nu-1}, v_{\nu-1}, w_{\nu-1}) \\
&h_{\nu,\nu-1,\nu-1} u_\nu = \mathcal{A} \cdot (v_{\nu-1}, w_{\nu-1})_{2,3} - U_{\nu-1} h_u \\
&h_v = \mathcal{A} \cdot (u_\nu, V_{\nu-1}, w_{\nu-1}) \\
&h_{\nu,\nu,\nu-1} v_\nu = \mathcal{A} \cdot (u_\nu, w_{\nu-1})_{1,3} - V_{\nu-1} h_v \\
&h_w = \mathcal{A} \cdot (u_\nu, v_\nu, W_{\nu-1}) \\
&h_{\nu\nu\nu} w_\nu = \mathcal{A} \cdot (u_\nu, v_\nu)_{1,2} - W_{\nu-1} h_w
\end{aligned}
$$

- **end**

Approximate

$$
\mathcal{A} \approx (U_m, V_m, W_m) \cdot \mathcal{H}, \qquad \mathcal{H} = \left(U_m^T, V_m^T, W_m^T\right) \cdot \mathcal{A}
$$

Tensor Krylov Methods

- Many variants are possible: see the talk by Berkant Savas in the session **MS117** Friday at 4.30
- Suitable for
    - sparse tensors
    - tensors whose dimensions vary rapidly (new data)

Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints

Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence

Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence
- Sparse tensors: Krylov methods

Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence
- Sparse tensors: Krylov methods
- Many fundamental mathematical and algorithmic problems remain

Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence
- Sparse tensors: Krylov methods
- Many fundamental mathematical and algorithmic problems remain
- Numerous new applications in information sciences

Conclusions

- Tensor methods/algorithms without index-wrestling
  - Indices hidden using matrix-inspired notation and object-oriented software
  - Generalization to higher order tensors is straightforward
  - Partial contractions play the role of adjoints
- Grassmann optimization (for Tucker model)
  - Needed because tensors cannot be deflated like matrices
  - Unconstrained optimization
  - Newton: Quadratic convergence
- Sparse tensors: Krylov methods
- Many fundamental mathematical and algorithmic problems remain
- Numerous new applications in information sciences
- Tensor algorithms and computations can be (easily) managed if we define the right abstractions!

Short Bibliograhy I

J. D. Carroll and J. J. Chang.
Analysis of individual differences in multidimensional scaling via an n-way generalization of Eckart-Young decomposition.
Psychometrika, 35:Psychometrika, 1970.

L. De Lathauwer, B. De Moor, and J. Vandewalle.
A multilinear singular value decomposition.
SIAM J. Matrix Anal. Appl., 21:1253–1278, 2000.

L. De Lathauwer, B. De Moor, and J. Vandewalle.
On the best rank-1 and rank-$(R_1, R_2, \ldots, R_N)$ approximation of higher-order tensor.
SIAM J. Matrix Anal. Appl., 21:1324–1342, 2000.

V. de Silva and L.-H. Lim.
Tensor rank and the ill-posedness of the best low-rank approximation problem.
SIAM J. Matrix Anal. Appl., to appear, 2007.

Daniel M. Dunlavy, Tamara G. Kolda, and W. Philip Kegelmeyer.
Multilinear algebra for analyzing data with multiple linkages.
Technical Report SAND2006-2079, Sandia National Laboratories, Albuquerque, NM and Livermore, CA, April 2006.

Short Bibliograhy II

L. Eldén and B. Savas.
A Newton–Grassmann method for computing the best multi-linear rank-(r_1, r_2, r_3) approximation of a tensor.
Technical Report LITH-MAT-R-2007-6-SE, Department of Mathematics, Linköping University, 2007.
Submitted to SIMAX.

R. Harshman.
Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis.
Technical Report 16:1-84, UCLA working papers in phonetics, 1970.

T. Kolda, B. Bader, and J. Kenny.
Higher-order web link analysis using multilinear algebra.
In *Proc. 5th IEEE International Conference on Data Mining, ICDM05*, pages 27–30. IEEE Computer Society Press, 2005.

T. G. Kolda and B. W. Bader.
Tensor decompositions and applications.
SIAM Review, 50:??, 2008, to appear.

B. Savas and L. Eldén.
Handwritten digit classification using higher order singular value decomposition.
Pattern Recognition, 40:993–1003, 2007.

Short Bibliograhy III

A. Smilde, R. Bro, and P. Geladi.
Multi-way Analysis: Applications in the Chemical Sciences.
Wiley, 2004.

L. R. Tucker.
The extension of factor analysis to three-dimensional matrices.
In H. Gulliksen and N. Frederiksen, editors, *Contributions to Mathematical Psychology*, pages 109–127. Holt, Rinehart and Winston, New York, 1964.

L. R. Tucker.
Some mathematical notes on three-mode factor analysis.
Psychometrika, 31:279–311, 1966.