Numerical tensor methods and their applications

I.V. Oseledets

2 May 2013

What is this course is about

This course is mostly on numerical methods of linear algebra in multilinear settings.

What is this course is about

This course is mostly on numerical methods of linear algebra in multilinear settings.

Goal: develop universal tools for working with high-dimensional problems.

All lectures

4 lectures,

- 2 May, 08:00 - 10:00: Introduction: ideas, matrix results, history.
- 7 May, 08:00 - 10:00: Novel tensor formats (TT, HT, QTT).
- 8 May, 08:00 - 10:00: Advanced tensor methods (eigenproblems, linear systems).
- 14 May, 08:00 - 10:00: Advanced topics, recent results and open problems.

Lecture 1

- Motivation
- Matrix background
- Canonical and Tucker formats
- Historical overview

Motivation

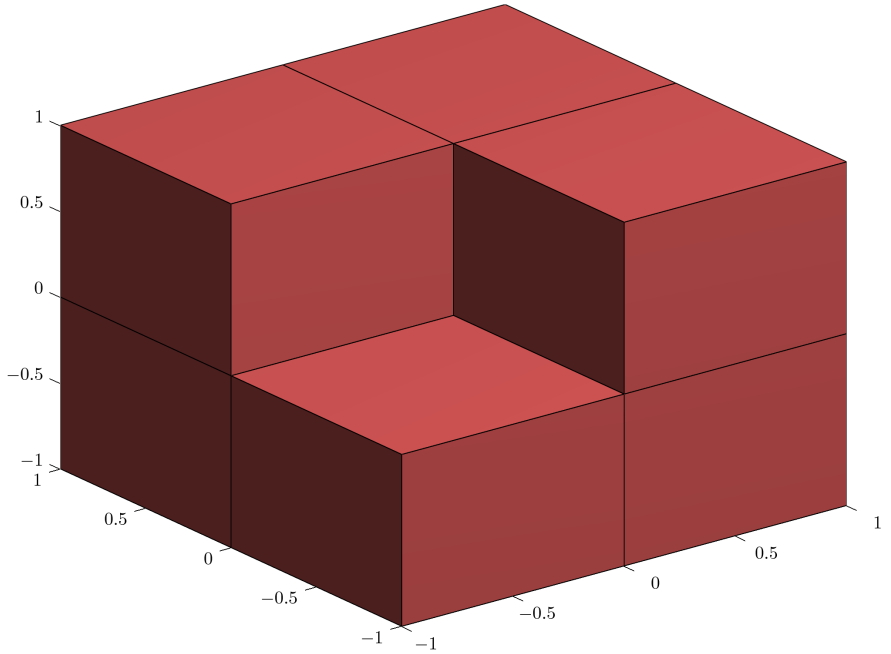
Main points

- High-dimensional problems appear in diverse applications
- Standard methods do not scale well in many dimensions

Motivation

Solution of high-dimensional differential and integral equations on fine grids

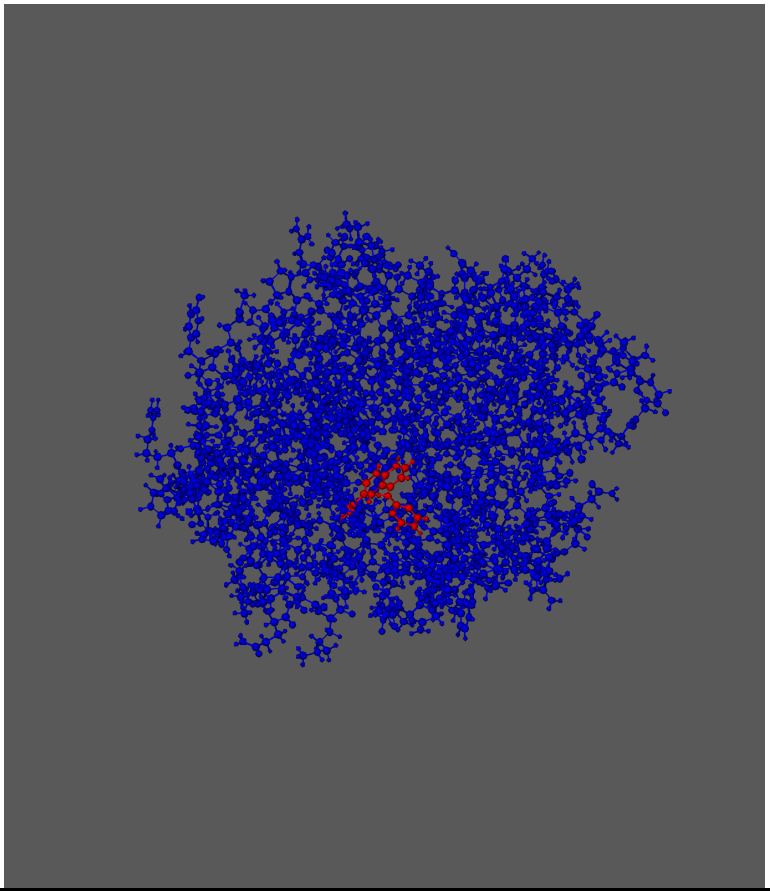
Typical cost: $\mathcal{O}(N^3) \longrightarrow \mathcal{O}(N)$ or even $\mathcal{O}(\log^{\alpha} N)$.



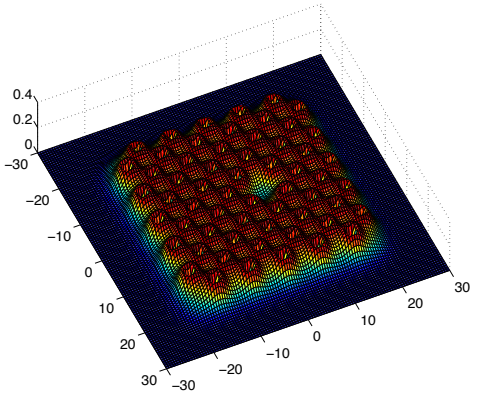
Motivation

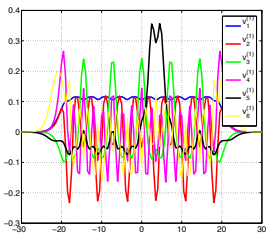
Ab initio computations and computational material design

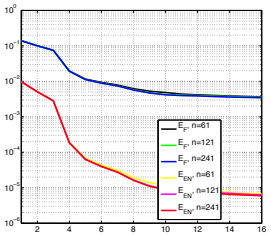
Protein-ligand docking (D. Zheltkov)



Density functional theory for large clusters (V. Khoromskaia)

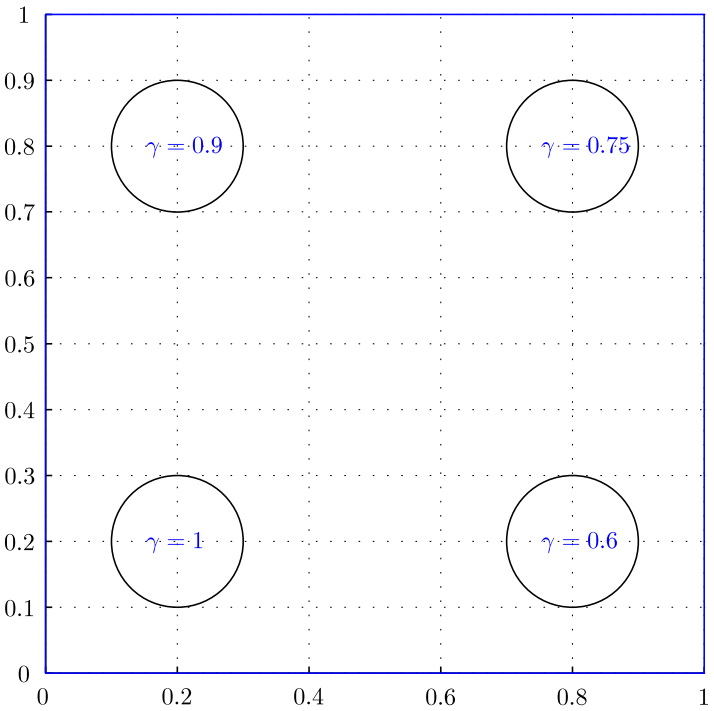






Motivation

Construction of reduced order models for multiparametric/stochastic systems in engineering



Diffusion problem

$\nabla a(p)\Delta u = f(p),$

$p = (p_1, p_2, p_3, p_4)$

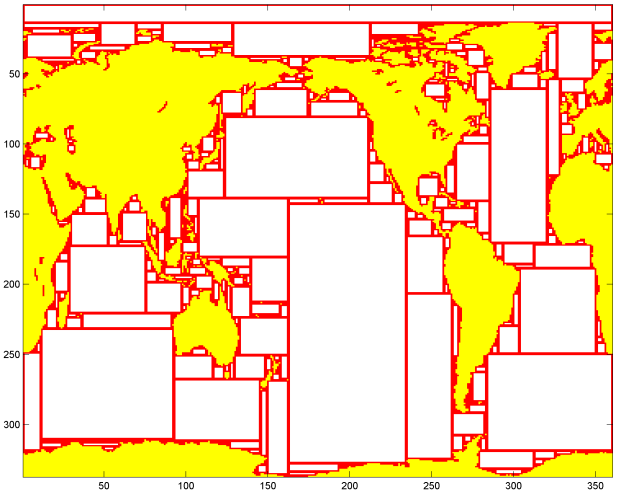
Approximate u using only few snapshots.

Motivation

Data mining and compression

Images

Computational data (temperature)



Why tensors are important

The multivariate functions are related to the multivariate arrays, or tensors:

Why tensors are important

The multivariate functions are related to the multivariate arrays, or tensors:

Take a function: $f(x_1, \ldots, x_d)$

Take tensor-product grid

Get a tensor:

$$A(i_1, \ldots, i_d) = f(x_1(i_1), \ldots, x_d(i_d))$$

Literature

- T. Kolda and B. Bader, Tensor decompositions and applications, SIREV (2009)
- W. Hackbusch, Tensor spaces and numerical tensor calculus, 2012
- L. Grasedyck, D. Kressner, C. Tobler, A literature survey of low-rank tensor approximation techniques, 2013

Software

Some software will be used:

- Tensor Toolbox 2.5 (T. Kolda)
- TT-Toolbox (http://github.com/oseledets/TT-Toolbox)

There is also a Python version (http://github.com/oseledets/ttpy) which has similar functionality now.

Where tensors come from

- d-dimensional PDE: $\Delta u = f$, $u = u(x_1, \ldots, x_d)$
- PDE with M parameters: $A(p)u(p) = f(p)$, $u = u(x, p_1, \ldots, p_M)$
- Data (images, video, hyperspectral images)
- Latent variable models, joint probability distributions
- Factor models
- Many others

Definitions

A tensor is a d-dimensional array:

$$A(i_1, \ldots, i_d), \quad 1 \leq i_k \leq n_k$$

Mathematically more correct definition:

Tensor is a polylinear form.

Definitions

Tensors form a linear vector space.

The natural norm is the Frobenius norm:

$$\|A\| = \sqrt{\sum_{i_1,\ldots,i_d} |A(i_1,\ldots,i_d)|^2}$$

Curse of dimensionality

Curse of dimensionality: Storage of a d-tensor with mode sizes n requires n^d elements.

Basic questions

- How to break the curse of dimensionality?
- How to perform (multidimensional) sampling?
- How to do everything efficiently and in a robust way?

Real-life problems

If you really need to compute something high-dimensional

, there is usually a way:

- Monte Carlo
- Special basis sets (radial basis functions)
- Best N-term approximations (wavelets, sparse grids)

But we want algebraic techniques...

Separation of variables

One of the few fruitful ideas is the idea of separation of variables

What is separation of variables

Separation rank 1:

$$f(x_1, \ldots, x_d) = u_1(x_1) u_2(x_2) \ldots u_d(x_d),$$

More general:

$$f(x_1, \ldots, x_d) \approx \sum_{\alpha=1}^{r} u_1(x_1, \alpha) \ldots u_d(x_d, \alpha).$$

Analytical examples

How to compute separated representations?

Analytical expressions (B. N. Khoromskij and many others):

$f(x_1, \ldots, x_d) = \frac{1}{x_1 + \ldots + x_d}$ based on the identity

$$\frac{1}{x} = \int_0^\infty \exp(-px)\, dp$$

$$r = \log \varepsilon^{-1} \log \delta^{-1}$$

Numerical computation of separated representations

We can try to compute the separated decomposition numerically.

How do we do that?

Canonical format

Tensors:

Canonical format:

$$A(i_1, \ldots, i_d) \approx \sum_{\alpha=1}^{r} U_1(i_1, \alpha) \ldots U_d(i_d, \alpha)$$

What happens in $d = 2$?

Two-dimensional case

$$A(i_1, i_2) \approx \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha)$$

Two-dimensional case

$$A(i_1, i_2) \approx \sum_{\alpha=1}^{r} U_1(i_1, \alpha) U_2(i_2, \alpha)$$

Matrix form: $A \approx UV^\top$,

Where U is $n \times r$, V is $m \times r$

Approximate rank-r approximation

SVD: definition

The fabulous SVD (singular value decomposition):

Every matrix can be represented as a product

$A = USV^*$, where U, V are orthonormal, S is a diagonal matrix with singular values $\sigma_i \geq 0$ on the diagonal.

SVD: complexity

Complexity of the SVD is $\mathcal{O}(n^3)$ (too much to compute $\mathcal{O}(nr)$ decomposition)

SVD: complexity

Complexity of the SVD is $\mathcal{O}(n^3)$ (too much to compute $\mathcal{O}(nr)$ decomposition)

Are there faster algorithms?

Skeleton decomposition

Yes: based on the skeleton decomposition

$$A \approx C\widehat{A}^{-1}R,$$

C — r columns of A, R — r rows of A, $\widehat{A}$ — submatrix on the intersection.

- Ex.1: Prove it
- Ex.2: Have you met skeleton dec. before?

Maximum volume principle

What happens if the matrix is of approximate low rank?

$$A \approx R + E, \quad \mathbf{rank}\, R = r, \quad \|E\|_C = \varepsilon$$

Maximum volume principle

Select the submatrix $\widehat{A}$ such that volume is maximal
(volume = absolute value of the determinant)

$$\|A - C\widehat{A}^{-1}R\| \leq (r+1)^2\varepsilon$$

Proof

E. E. Tyrtyshnikov, S.A. Goreinov, On quasioptimality of skeleton approximation of a matrix in the Chebyshev norm, doi: 10.1134/S1064562411030355

$$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix},$$

$$H = A - C\widehat{A}^{-1}R = A - \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix} A_{11}^{-1} \begin{pmatrix} A_{11} A_{21} \end{pmatrix}$$

Need: $|h_{ij}| \leq (r+1)^2 \delta_{r+1}(A)$

Proof

$$Z = \begin{pmatrix} A_{11} & v \\ u^\top & a_{ij} \end{pmatrix}$$

Entry h_{ij} can be found from:

$$\begin{pmatrix} I & 0 \\ -u^\top A_{11}^{-1} & 1 \end{pmatrix} Z = \begin{pmatrix} A_{11} & v \\ 0 & h_{ij} \end{pmatrix}$$

$$\det Z = h_{ij} \det A_{11}$$

Therefore,

$$|h_{ij}^{-1}| = \|Z^{-1}\|_C,$$

$$|h_{ij}| \leq (r+1)\sigma_{r+1}(Z)$$

Proof

Finally,

$$\sigma_{r+1}(Z) = \min_{U_Z, V_z} \|Z - U_Z V_Z^\top\|_2 \leq$$
$$(r+1)\|Z - U_Z V_Z^\top\|_C \leq (r+1)\delta_{r+1}(A)$$

Maxvol algorithm(1)

Ok, then, how to find a good submatrix?

Crucial algorithm: Maxvol submatrix in a $n \times r$ matrix.

Characteristic property: A is $n \times r$,

$$A\widehat{A}^{-1} = \begin{pmatrix} I \\ Z \end{pmatrix}, \quad |Z|_{ij} \leq 1.$$

Maxvol algorithm(2)

Problem: find maximal volume $r \times r$ submatrix in an $n \times r$ matrix.

Maxvol algorithm(2)

Problem: find maximal volume $r \times r$ submatrix in an $n \times r$ matrix.

Maxvol algorithm:

- Take some rows, put them in the first r.
- Compute $B = A\widehat{A}^{-1}$
- $B = \begin{pmatrix} I \\ Z \end{pmatrix}$
- Suppose maximal element in Z is in position (i, j).
- Swap i-th row with j-th row.
- Stop if maximal element is less than $(1 + \delta)$.

Maxvol algorithm(2)

Problem: find maximal volume $r \times r$ submatrix in an $n \times r$ matrix.

For an $n \times m$ matrix:

Find maximal volume in rows, then find maximal volume in columns

Ex. Implement an algorithm that searches for a maxvol submatrix.

Maxvol algorithm (demo)

Let us see how maxvol works...

Cross approximation

A typical scheme we use is the cross approximation approach, which uses **minimal information** from the matrix.

Cross approximation

1. $k = 0$, Select j_0, $U_0 = 0, V_0 = 0$.
2. Compute j_k-th column of the remainder $A_k = A - U_k V_k^\top$.
3. Find maximal element i_k in it, compute i_k-th row, compute maximal element $j_{k+1} \neq j_k$.
4. Compute the next cross: $u_k = A_k e_{j_k}$, $v_k = A_k^\top e_{i_k}$, $u_k = u_k / A_k(i_k, j_k)$, $U_k = [U_{k-1}, u_k]$, $V_k = [V_{k-1}, v_k]$.
5. If $\|u_k v_k^\top$ is small, stop, else go to 1.

Randomized techniques

Randomized techniques for low-rank approximation became popular recently.

Sublinear randomized algorithms for skeleton decompositions, Jiawei Chiu and Laurent Demanet, http://arxiv.org/abs/1110.4193v2

Theorem

Let $A = USV^\top$ and U and V are μ-coherent, i.e. $\|U\|_C \leq \sqrt{\frac{\mu}{n}}$

Then, with high probability, one has to sample $I = \mu r \log n$ columns and rows uniformly, to get a $\mathcal{O}(\sigma_{r+1})$ bound.

What is the best cross algorithm?

I strongly believe, that the "best" cross algorithm is still to be found

And it is very important in higher dimensions!

Going to higher dimensions

How to generalize the idea of separation of variables to higher dimensions?

- SVD is good
- Best approximation exists
- Interpolation via skeleton

Canonical format (2)

$$A(i_1, \ldots, i_d) \approx \sum_{\alpha=1}^{r} U_1(i_1, \alpha) \ldots U_d(i_d, \alpha)$$

r is called (approximate) canonical rank, U_k — canonical factors.

Canonical format(3)

Good things about the canonical format:

- Low number of parameters dnr
- Uniqueness results (Kruskal theorem)

Canonical format(3)

Let A be a 3-tensor with (U, V, W) canonical decomposition of rank R,

, and $k(U) + k(V) + k(W) \geq 2R + 3$, then the decomposition is unique.

$k(X)$ — Kruskal rank (spark in compressed sensing),

Def: $k(X) + 1$ is the minimal number of linearly dependent columns in X.

Proof is highly nontrivial (Est time: ~1.5 lectures!)

Canonical format (4)

Bad things about the canonical format:

- Best approximation may not exist
- Canonical rank is NP-complete (matrix rank is ...)
- No good algorithm

Bad example (1)

$$f(x_1, \ldots, x_d) = x_1 + x_2 + \ldots x_d,$$

Canonical rank d (no proof is known), can be approximated with rank-2 with any accuracy!

Bad example (2)

Canonical rank may depend on the field (matrix rank can not!)

$$f(x_1, \ldots, x_d) = \sin(x_1 + \ldots + x_d)$$

- Complex field: 2
- Real field: d (Ex.: prove it)

Alternating least squares

The main algorithm for the computation of the canonical decomposition is the Alternating Least Squares (ALS) algorithm.

- Easy to implement
- Known for its very slow convergence (swaps)
- Local convergence proven only recently (Uschmajew, A.)

Alternating least squares

Treat approximation as an optimization problem:

$$\|A - (U, V, W)\|_F \to \min$$

Three steps:

1. Fix V, W, update U (linear least squares)
2. Fix U, W, update V
3. Fix U, V, update W.

Exercise: write down comp. formula and implement them.

Example from the complexity theory

There are cases, where the canonical format comes from a model:

Matrix multiplication:

Example from the complexity theory

There are cases, where the canonical format comes from a model:

Matrix multiplication:

$$C = AB,\ c = f(a, b),\ c_i = \sum_{ij} E_{ijk} a_j b_k$$

If the canonical rank of E is r, computation of C requires r multiplications

- $2 \times 2 : 4 \times 4 \times 4$ tensor, rank 7 (Strassen)
- $3 \times 3 : 9 \times 9 \times 9$ tensor, rank is unknown $19 \leq r \leq 23$.

Example from the complexity theory

There are cases, where the canonical format comes from a model:

Matrix multiplication:

$$C = AB,\ c = f(a, b),\ c_i = \sum_{ij} E_{ijk} a_j b_k$$

If the canonical rank of E is r, computation of C requires r multiplications

- $2 \times 2 : 4 \times 4 \times 4$ tensor, rank 7 (Strassen)
- $3 \times 3 : 9 \times 9 \times 9$ tensor, rank is unknown $19 \leq r \leq 23$.

It is fascinating.

What about sampling?

Can we generalize skeleton decomposition?

What about sampling?

Can we generalize skeleton decomposition?

No

Try it yourself: a simple generalization of a “cross”.

Another attempt: Tucker

Another attempt to avoid was the Tucker format

(Tucker 1966, Lathauwer, 2000+)

$$A(i,j,k) \approx$$
$$\sum_{\alpha\beta\gamma} G(\alpha,\beta,\gamma) U_1(i,\alpha) V(j,\alpha) W(k,\alpha)$$

Tucker and SVD

You can compute Tucker by means of the SVD:

- Compute unfoldings: A_1, A_2, A_3
- Compute left SVD factors: $A_i \approx U_i \Phi_i$
- Compute the core: $G = A \times_1 U_1^\top \times_2 U_2^\top \times_3 U_3^\top$.

Tucker and the cross

You can generalize skeleton to Tucker (O., Savostyanov, Tyrtyshnikov, 2008)

Compute good columns in A_i, find core by interpolation.

Problem with the Tucker format

Q: What is the main problem with the Tucker format?

Problem with the Tucker format

Q: What is the main problem with the Tucker format?

A: Curse of dimensionality

The core takes r^d elements!

Summary

What we have?

- Canonical format: low number of parameters, no algorithms
- Tucker format: SVD-based algorithms, the curse

Main algebraic question

Can we find something inbetween?

Lecture 2

- The Tree-Tucker, Tensor Train, Hierarchical Tucker formats
- Their difference
- Concept of Tensor Networks
- Stability and quasioptimality
- Basic arithmetic (with illustration)
- Cross approximation formula (with illustrations)
- QTT-format (part 1)

Lecture 3

- QTT-format (part 2), application to numerical integration
- QTT-Fourier transform and its relation to tensor networks
- QTT-convolution, explicit representation of Laplace-like tensors
- DMRG/AMEN techniques
- Solution of linear systems in the TT-format
- Solution of eigenvalue problems in the TT-format

Lecture 4

Advanced topics: New applications, recent results and open problems

- Solution of non-stationary problems
- Global optimization via the TT-cross
- Latent variable models (finance and natural language processing)
- Approximation results in quantum information theory (Hastings area law)
- Open problems