Lecture1

Vika Chekalina (@sayankotor)

based on Skolkovo CDISE Matrix and Tensor Factorization Course, Caltech Tensor&Neural Network Course

23 sent 2020

Intro

Tensor

- Tensor - multidimensional array:
  - multi-channel images
  - time series
  - data, obtained from experiments under various conditions
  - data describing objects in different sensors' view
- Initially, every dimension make it's own sense: time, channel, mode.

Curse of dimensionality

The phenomenon whereby the number of elements of an Nth-order tensor grows exponentially with the tensor order, N. Tensor volume be very high, thus requiring enormous computational and memory resources to process such data.

Challenges

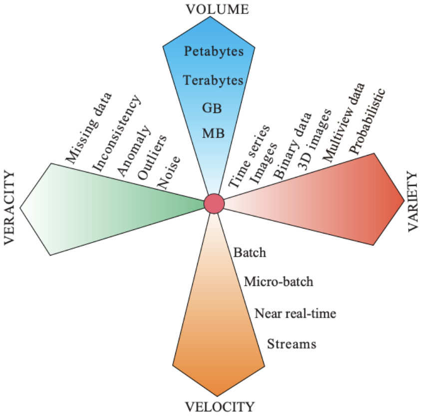


Figure: The 4V challenges for big data

Big data are: too big, noisy, slow to process, different types

Aprroximation

- Initially, every dimension make it's own sense: time, channel, mode.
- Tensors may contain redundant information: excessive degrees of freedom can have the inherent dependencies among the each other (example: data under different conditions when conditions can't influence on objects vastly.
- Handling of the inherent dependencies leeds to compression this representation

Aprroximation

Compression of multidimensional large-scale data

N-variate function $f(x) = f(x_1, x_2, x_3, ..., x_N)$ can be represented as
$f(x) \approx f^{(1)}(x_1) \cdot f^{(2)}(x_2) \cdot f^{(3)}(x_3) \cdot ... \cdot f^(N)(x_N)$

- Descritization of $f(x)$ is a N-dimensional tensor
- Approximation and compression leeds to extraction the meaningful information without noise and extra dependencies
- Approximation by several low-rank objects simplifies the problem of storing.

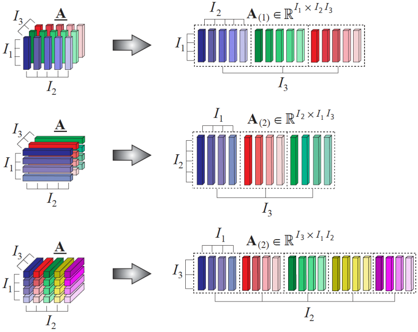
Casting

- Tenzor $\hat{X}$, matrix M, vector $\vec{y}$
- Strange special symbol $\hat{1}_R^T$: $\sum_{i=1}^{R} y_i = \hat{1}_R^T \vec{y}$
- Outer product $\circ$
- Kronecker product $\otimes$
- Hadamarad product $*$
- Khartri-rao product $\odot$
- Mode-n matricization (changing the number of dimensions) $\hat{X}_{(n)}$
- Vectorization (changing the number pf dimensions)
- Frobenious norm $\|\hat{X}\|_F = \sum_{i_1=1} \cdots \sum_{i_N=1} x_{i_1 \dots i_N}$

Matricization, Unfolding

Tensor unfolding, or matrization, is a fundamental operation and a building block for most tensor methods. Considering a tensor as a multi-dimensional array, unfolding it consists of reading its element in such a way as to obtain a matrix instead of a tensor.

$$\hat{X} \in \mathbb{R}^{I_1 \times \cdots \times I_n \times \cdots \times I_N} \rightarrow X_{(n)} \in \mathbb{R}^{I_n \times I_1 \cdot \cdots \cdot I_N}$$



Mode-1, mode-2, and mode-3 matricizations of a 3rd-order tensor

Matricization, Unfolding

Two way

- Kolda and Bader, Fortran memory order (column-major):
  A[0][0] A[1][0] A[2][0] A[0][1] A[1][1] A[2][1] A[0][2] A[1][2] A[2][2]
- Tensorly, C memory Order (row-major):
  A[0][0] A[0][1] A[0][2] A[1][0] A[1][1] A[1][2] A[2][0] A[2][1] A[2][2]

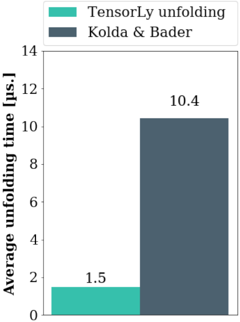


Figure: Avarege accross the modes of the unfolding time for $(100, 10, 15, 10, 100)$ sized tensor

Matricization, Unfolding

Kolda and Bader

Maps element $(i_1 \times i_2 \times \cdots \times i_N)$ to (i_n, j), where

$$j = \sum_{k=0, k\neq n}^{N} \left[i_k \prod_{m=0, m\neq n}^{k-1} I_m \right]$$

$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \quad X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$$X_{[0]} = \begin{bmatrix} 0 & 2 & 4 & 6 & 1 & 3 & 5 & 7 \\ 8 & 10 & 12 & 14 & 9 & 11 & 13 & 15 \\ 16 & 18 & 20 & 22 & 17 & 19 & 21 & 23 \end{bmatrix}$$

$$X_{[1]} = \begin{bmatrix} 0 & 1 & 8 & 9 & 16 & 17 \\ 2 & 3 & 10 & 11 & 18 & 19 \\ 4 & 5 & 12 & 13 & 20 & 21 \\ 6 & 7 & 14 & 15 & 22 & 23 \end{bmatrix}$$

Matricization, Unfolding

Tensorly

Maps element $(i_1 \times i_2 \times \cdots \times i_N)$ to (i_n, j), where $j = \sum_{k=0, k \neq n}^{N} i_k \times \prod_{m=k+1}^{N} I_m$

$$X_0 = \begin{bmatrix} 0 & 2 & 4 & 6 \\ 8 & 10 & 12 & 14 \\ 16 & 18 & 20 & 22 \end{bmatrix} \; X_1 = \begin{bmatrix} 1 & 3 & 5 & 7 \\ 9 & 11 & 13 & 15 \\ 17 & 19 & 21 & 23 \end{bmatrix}$$

$$X_{[0]} = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 \\ 16 & 17 & 18 & 19 & 20 & 21 & 22 & 23 \end{bmatrix}$$

$$X_{[1]} = \begin{bmatrix} 0 & 8 & 16 & 1 & 9 & 17 \\ 2 & 10 & 18 & 3 & 11 & 19 \\ 4 & 12 & 20 & 5 & 13 & 21 \\ 6 & 14 & 22 & 7 & 15 & 23 \end{bmatrix}$$

Matricization, Unfolding

Properties

Tensor $\hat{X} \in R^{I_1 \times \cdots \times I_n}$ is decomposed into $\left[\hat{G}; U^(1) \cdots U^(n)\right]$

- Kolda and Bader: reverse order
  $X_{[n]} = U^{(n)} G_{[n]} U^{(N)} \ldots U^{(n+1)} \otimes U^{(n-1)} \ldots \otimes U^{(1)}$
- Tensorly: direct order
  $X_{[n]} = U^{(n)} G_{[n]} U^{(1)} \ldots U^{(n-1)} \otimes U^{(n+1)} \ldots \otimes U^{(N)}$

Vectorization

Isomorphism that maps the element of tensor to vector:

$\mathbb{R}^{I_1 \times \cdots \times I_N} \rightarrow (I_1 \times \cdots \times I_N)$

$$j = \sum_{k=0}^{N} i_k \prod_{m=k+1}^{N} I_m = i_N + i_{N-1} I_N + i_{N-2} I_N I_{N-2} + \cdots + i_1 I_2 \ldots I_N$$

or

$$j = \sum_{k=0}^{N} \left[i_k \prod_{m=0}^{k-1} I_m \right] = i_1 + i_2 I_1 + i_3 I_1 I_2 + i_N I_1 \ldots I_N$$

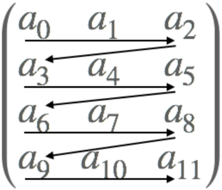


Figure: Numpy, Tensorly

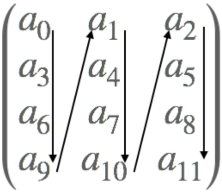


Figure: Matlab tools

Tensor reshaping operations

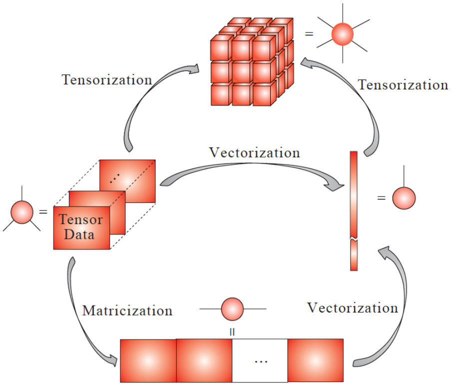


Figure: Tensor reshaping operations: Matricization, vectorization and tensorization. Matricization refers to converting a tensor into a matrix, vectorization to converting a tensor or a matrix into a vector, while tensorization refers to converting a vector, a matrix or a low-order tensor into a higher-order tensor.

Outer product

Outer product of tensors N-order tensor $\hat{A} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ and M-order tensor $\hat{A} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_M}$ is a (N+M) order tensor $\hat{C}$:

$$c_{i_1 \dots i_N j_1 \dots j_M} = a_{i_1 \dots i_N} b_{j_1 \dots j_M} \tag{1}$$

- outer product change the number of dimensions, not dimensions!
- Outer product of $\vec{a}$, $\vec{b}$, $\vec{c}$ forms a tensor X, with entries $x_{ijk} = a_i \cdot b_j \cdot c_k$
- what is the rank of this tensor?

Mode-n product

Mode-n product of tensor $\hat{A} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \cdots \times I_n \times I_N}$ and matrix $B \in \mathbb{R}^{J \times I_n}$ tensor $\hat{C} \in \mathbb{R}^{I_1 \times I_2 \times I_3 \cdots \times J \cdots \times I_N}$, that has elements:

$$c_{i_1 \ldots i_{n-1} j i_{n+1} \ldots j_N} = \sum_{i_n} a_{i_1 \ldots i_n \ldots i_N} b_{j i_n} \tag{2}$$

- n-mode product change the size of the corresponding tensor along the dimension n!
- n-mode product is the case of 3-d tensor is like matrix-matrix multiplication, where the second matrix is n-mode unfolding:

$$\hat{X} \times_n M = M \hat{X}_{[n]} \tag{3}$$

Kronecker product, left Kronecker product

For an $A \in \mathbb{R}^{I \times J}$ matrix a $B \in \mathbb{R}^{K \times L}$, the standard (Right) Kronecker product, $A \otimes_L B$, and the Left Kronecker product, $A \otimes_L B$, are the following $\mathbb{R}^{IK \times JL}$ matrices

$$C = \begin{bmatrix} a_{11}B & a_{12}B & \dots \\ \vdots & \ddots & \\ a_{J1}B & & a_{IJ}B \end{bmatrix}$$

$$C_L = \begin{bmatrix} b_{11}A & b_{12}A & \dots \\ \vdots & \ddots & \\ a_{L1}A & & b_{KL}A \end{bmatrix}$$

Kronecker product, left Kronecker product

Properties

- $(A \otimes B)^T (C \otimes D) = (A^T C) \otimes (B^T D)$
- $(A \otimes B)(E \odot F) = (AE) \odot (BF)$

Khartri-Rao product

Khartri-Rao product

- Given matrix $A \in \mathbb{R}^{I \times R}$ and $B \in \mathbb{R}^{J \times R}$
- $\mathrm{A} \odot \mathrm{B} = [\mathrm{a}_1 \otimes \mathrm{b}_1, \mathrm{a}_2 \otimes \mathrm{b}_2, \ldots, \mathrm{a}_k \otimes \mathrm{b}_k] \in \mathbb{R}^{IJ \times R}$

Properties

- $(\mathrm{A} \odot \mathrm{B})^T(\mathrm{A} \odot \mathrm{B}) = A^T A * B^T B$
- $(\mathrm{A} \odot \mathrm{B})^\dagger = ((A^T A) * (B^T B))^{-1}(\mathrm{A} \odot \mathrm{B})^T$

Hadamarad Product

Hadamarad Product $*$

- Elementwise product of the objects with the same order and the same size

Some vectorization properties

- $\mathrm{vec}\left(\mathbf{A}\mathbf{B}^T\right) = (\mathbf{B} \odot \mathbf{A})\mathbf{1}_R$
- $\mathrm{vec}\left(\mathbf{A}\,\mathrm{diag}(\boldsymbol{s})\mathbf{B}^T\right) = (\mathbf{B} \odot \mathbf{A})\boldsymbol{s}$
- $\mathrm{vec}\left(\mathbf{A}\mathbf{G}\mathbf{B}^T\right) = (\mathbf{B} \otimes \mathbf{A})\,\mathrm{vec}(\mathbf{G})$
- $\mathrm{vec}(\mathbf{A} \circledast \mathbf{B}) = \mathrm{vec}(\mathbf{A}) \circledast \mathrm{vec}(\mathbf{B})$
- $\mathrm{vec}(\mathbf{A}\mathbf{B} - \mathbf{B}\mathbf{A}) = (\mathbf{I} \otimes \mathbf{A} - \mathbf{A}^T \otimes \mathbf{I})\,\mathrm{vec}(\mathbf{B})$

CP decomposition

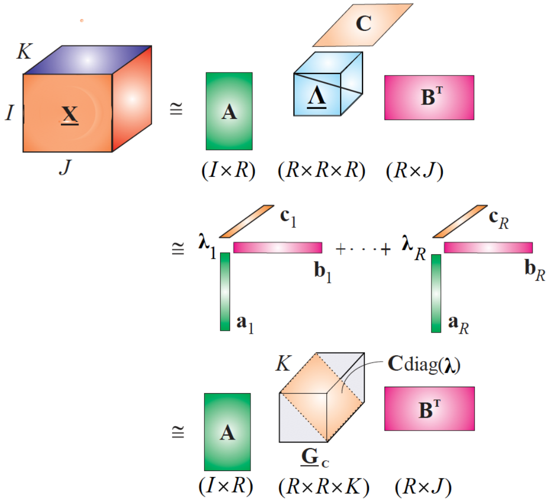
Kruskal Tensor

Kruskal tensor - tensor which be expressed as a finite sum of rank-1 tensors, in the form

$$\hat{X} = \sum_{r=1}^{R} \lambda_r \vec{b_1} \circ \vec{b_2} \cdots \circ \vec{b_n} \tag{4}$$

It also known under the names of CANDECOMP / PARAFAC, Canonical Polyadic Decomposition (CPD), or simply the CP decomposition.

CP decomposition



CP decomposizition

Tensor Rank

The tensor rank, also called the CP rank, is a natural extension of the matrix rank and is defined as a minimum number, R, of rank-1 terms in an exact CP decomposition of the form in

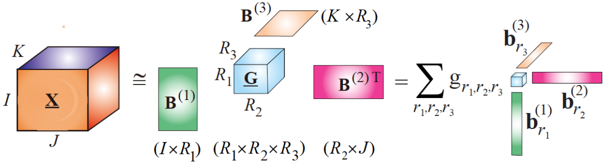
Best Decomposition

- If we define Norm $||||$ function, the best rank-r approximation is argmin $||Y - \hat{Y}||$
- For tensors of dimensions $>=$ 3 this argmin can not exists
- therefore, not every tensor has the rank
- more in https://arxiv.org/pdf/math/0607647.pdf

Tucker decomposition

$$\hat{X} = \sum_{r_1=1}^{R_1} \sum_{r_N=1}^{R_N} g_{r1,r2..rn} \vec{b_1} \circ \vec{b_2} \cdots \circ \vec{b_n} \tag{5}$$

Scheme of the Tucker decomposition



Tucker and CP decompositions

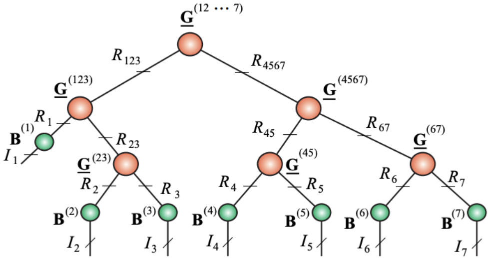
Properties of decomposition

CP	Tucker
Scalar product	
$x_{i_1,\dots,i_N} = \sum_{r=1}^{R} \lambda_r \, b^{(1)}_{i_1,r} \cdots b^{(N)}_{i_N,r}$	$x_{i_1,\dots,i_N} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\dots,r_N} \, b^{(1)}_{i_1,r_1} \cdots b^{(N)}_{i_N,r_N}$
Outer product	
$\underline{\mathbf{X}} = \sum_{r=1}^{R} \lambda_r \, \mathbf{b}^{(1)}_{r} \circ \cdots \circ \mathbf{b}^{(N)}_{r}$	$\underline{\mathbf{X}} = \sum_{r_1=1}^{R_1} \cdots \sum_{r_N=1}^{R_N} g_{r_1,\dots,r_N} \, \mathbf{b}^{(1)}_{r_1} \circ \cdots \circ \mathbf{b}^{(N)}_{r_N}$
Multilinear product	
$\underline{\mathbf{X}} = \underline{\boldsymbol{\Lambda}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$ $\underline{\mathbf{X}} = [\![\underline{\boldsymbol{\Lambda}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \dots, \mathbf{B}^{(N)}]\!]$	$\underline{\mathbf{X}} = \underline{\mathbf{G}} \times_1 \mathbf{B}^{(1)} \times_2 \mathbf{B}^{(2)} \cdots \times_N \mathbf{B}^{(N)}$ $\underline{\mathbf{X}} = [\![\underline{\mathbf{G}}; \mathbf{B}^{(1)}, \mathbf{B}^{(2)}, \dots, \mathbf{B}^{(N)}]\!]$
Vectorization	
$\mathrm{vec}(\underline{\mathbf{X}}) = \left(\bigodot_{n=N}^{1} \mathbf{B}^{(n)} \right) \boldsymbol{\lambda}$	$\mathrm{vec}(\underline{\mathbf{X}}) = \left(\bigotimes_{n=N}^{1} \mathbf{B}^{(n)} \right) \mathrm{vec}(\underline{\mathbf{G}})$
Matricization	
$\mathbf{X}_{(n)} = \mathbf{B}^{(n)} \boldsymbol{\Lambda} \left(\bigodot_{m=N,\, m\neq n}^{1} \mathbf{B}^{(m)} \right)^{\mathrm{T}}$ $\mathbf{X}_{<n>} = (\bigodot_{m=n}^{1} \mathbf{B}^{(m)}) \boldsymbol{\Lambda} (\bigodot_{m=N}^{n+1} \mathbf{B}^{(m)})^{\mathrm{T}},$	$\mathbf{X}_{(n)} = \mathbf{B}^{(n)} \mathbf{G}_{(n)} \left(\bigotimes_{m=N,\, m\neq n}^{1} \mathbf{B}^{(m)} \right)^{\mathrm{T}}$ $\mathbf{X}_{<n>} = (\bigotimes_{m=n}^{1} \mathbf{B}^{(m)}) \mathbf{G}_{<n>} (\bigotimes_{m=N}^{n+1} \mathbf{B}^{(m)})^{\mathrm{T}}$

HT and TT

Hierarchical Tucker

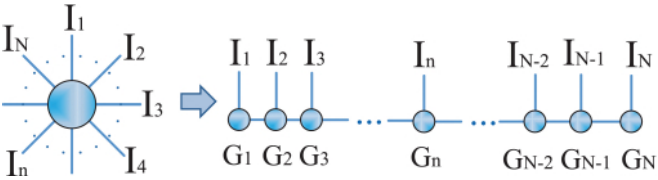
- when you represent decomposition as a tree
- each node correspond to a factor U_k
- factor can be decomposed too by it's own factors



HT and TT

Tensor Train (Matrix Product State)

The special case of Hierarchical Tucker



Tucker, CP and TT decompositions

- CP decomposition addresses the curse of dimensionality,
- Tucker is more stable, but has no sense if dimension is more than 5-6: core tensor has curse of dimensionality too.
- Tensor Train decomposition provides both very good numerical properties and the stable rank reduction to control the error of approximation

Unfolding of initial tensor via factors

Very important formulas!
These formulas are used in decomposition algorithms

Canonical Decomposition

$X = [|A, B, C|]$

- $X_{[0]} = A(B \odot C)^T$
- $X_{[1]} = B(A \odot C)^T$
- $X_{[1]} = C(A \odot B)^T$

Tucker

$X = [|G; A, B, C|]$

- $X_{[0]} = A(B \otimes C)^T$
- $X_{[1]} = B(A \otimes C)^T$
- $X_{[1]} = C(A \otimes B)^T$

ALS

We have tensor $\hat{T}$ and want to find it's CPD decomposition $\hat{T} = [|\hat{U}^{(1)}\hat{U}^{(2)}\hat{U}^{(3)}|]$, i.e find $\hat{U}^{(1)}, \hat{U}^{(2)}, \hat{U}^{(3)}$

- We can define every factor matrix from initial tensor $\hat{T}$ and another factor matrices
- We can obtain this definition from LLS(Linear least squares)
- On the zero step we initialize factor matrices
- Every step consists of number_of_factors substep
- For every substep we update one of factor matrices in assumption that another factor matrices is fixed on this activity
- Do until given number of iteration is reached, or given accuracy is reached

LLS

- $\min_X \|Y - X\beta\|^F$
- $\min_X (Y - X\beta)^T (Y - X\beta)$
- $Y^T Y - \beta^T X^T Y - Y^T X\beta + \beta^T X^T X\beta = 0$
- $X^T Y = X^T X\beta$
- $\beta = (X^T X)^{-1} X^T Y$
- $\beta^T = Y^T X (X^T X)^-1$
- $\beta^T = Y^T (X^T)^\dagger$, where $X^\dagger$ - pseudo-inverse (Moore–Penrose inverse matrix)

LLS

- $\|\hat{X}_{[0]} - U^{(0)}(U^{(1)} \odot U^{(2)})^T\|$
- $U^{(0)} = \hat{X}_{[0]}[(U^{(1)} \odot U^{(2)})^T]^\dagger$
- using properties, the pseudoinverse matrix can be expressed in terms of the already known objects

Example: norm of Kruskal Tensor

$$
\begin{aligned}
&vec(\hat{Y}) = (C \odot B \odot A)\hat{1}_R \\
&||\hat{Y}|| = vec(\hat{Y})^T vec(\hat{Y}) = \\
&= \hat{1}_R^T (C \odot B \odot A)^T (C \odot B \odot A)\hat{1}_R \\
&= \hat{1}_R^T (C^T C * B^T B * A^T A)\hat{1}_R
\end{aligned}
$$