Faculté des Sciences et Techniques
Applied Multilinear Algebra

Université de Limoges
M2 – Semester 1 – 2024/2025

CHAU Dang Minh

TENSOR DECOMPOSITIONS
THEORY AND IMPLEMENTATION

1 CP Decomposition

By convention, let $\mathbf{A} \in \mathbb{R}^{I \times J}$ be a matrix, we shall denote by $\mathbf{a}_j \in \mathbb{R}^I$, where $1 \leq j \leq J$ the j-th column of $\mathbf{A}$. Let $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ be an order-3 tensor. The CP decomposition seeks for matrices $\mathbf{A} \in \mathbb{R}^{I \times R}$, $B \in \mathbb{R}^{J \times R}$ and $C \in \mathbb{R}^{K \times R}$ such that

$$\mathcal{X} = \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{1}$$

such that R is the smallest possible number. In such case, R is called the *rank* of $\mathcal{X}$. It is usually useful to standardize the column vectors of the matrices to unit norm by introducing scaling factors $\boldsymbol{\lambda} = (\lambda_1, \ldots, \lambda_r)^\top$, i.e.

$$\mathcal{X} = \sum_{r=1}^{R} \lambda_r \, \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r. \tag{2}$$

The decomposition can be described briefly as $\mathcal{X} = [[\boldsymbol{\lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]$. Let $\Lambda = \operatorname{diag}(\lambda)$, we have the equivalent matricization expressions

$$\begin{cases} \mathbf{X}_{(1)} = \mathbf{A}\Lambda(\mathbf{C} \odot \mathbf{B})^\top \\ \mathbf{X}_{(2)} = \mathbf{B}\Lambda(\mathbf{C} \odot \mathbf{A})^\top \\ \mathbf{X}_{(3)} = \mathbf{C}\Lambda(\mathbf{B} \odot \mathbf{A})^\top. \end{cases} \tag{3}$$

There is no known finite algorithm for determining the rank of a tensor [1]. Therefore, for each $R = 1, 2, \ldots$, we attempt to minimize the difference between $\mathcal{X}$ and a rank-R tensor $\hat{\mathcal{X}} = [[\boldsymbol{\lambda}, \mathbf{A}, \mathbf{B}, \mathbf{C}]]$, i.e.

$$\min_{\hat{\mathcal{X}}} \|\mathcal{X} - \hat{\mathcal{X}}\|^2 \tag{4}$$

until the difference is less than a given tolerance. Using (3), we can write, for example, mode-1 matricization form of the problem as

$$\min_{\hat{\mathcal{X}}} \|\mathcal{X}_{(1)} - \mathbf{A}\Lambda(\mathbf{C} \odot \mathbf{B})\|. \tag{5}$$

The solution is

$$\mathbf{A}\Lambda = \mathcal{X}_{(1)}[(\mathbf{C} \odot \mathbf{B})^\top]^\dagger = \mathcal{X}_{(1)}(\mathbf{C} \odot \mathbf{B})(\mathbf{C}^\top\mathbf{C} \star \mathbf{B}^\top\mathbf{B})^\dagger \tag{6}$$

We use the last expression because it requires calculating the pseudoinverse of an $R \times R$ matrix rather than a $JK \times R$ matrix. Therefore, we can solve for each factor iteratively until a convergence criterion is met, particularly, the difference between the original tensor and the reconstructed tensor. The general case for N-way tensor is represented in the algorithm below.

Algorithm 1 CP Decomposition using Alternative Least Square

Input: $\mathcal{X} \in \mathbb{R}^{I_1, \ldots, I_N}$, $\epsilon = 1e - 10$, maxIter $= 5000$
  Initialize $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times R}$ for $n = 1, \ldots, N$ randomly
  $\boldsymbol{\lambda} \leftarrow \mathbf{0}_R$
  iter $\leftarrow 0$
  while $\|\mathcal{X} - [[\boldsymbol{\lambda}, \mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(n)}]]\| > \epsilon$ and iter $<$ maxIter **do**
    for $n = 1, \ldots, N$ **do**
      $\mathbf{V} \leftarrow \mathbf{A}_{(1)}^\top \mathbf{A}_{(1)} \star \ldots \star \mathbf{A}_{(n-1)}^\top \mathbf{A}_{(n-1)} \star \mathbf{A}_{(n+1)}^\top \mathbf{A}_{(n+1)} \star \ldots \star \mathbf{A}_{(N)}^\top \mathbf{A}_{(N)}$
      $\mathbf{A}_{(n)} \leftarrow \mathbf{X}_{(n)}(\mathbf{A}_{(N)} \odot \ldots \odot \mathbf{A}_{(n+1)}\mathbf{A}_{(n-1)} \odot \ldots \odot \mathbf{A}_{(1)})\mathbf{V}^\dagger$
      $\boldsymbol{\lambda} \leftarrow (\mathbf{a}_{(n)1}, \ldots, \mathbf{a}_{(n)r})^\top$
      Normalize columns of $\mathbf{A}_{(N)}$
    end for
  end while
Output: $\boldsymbol{\lambda}, \mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(N)}$.

2 Tucker Decompositions: HOSVD and HOOI

Let $R_n = \text{rank}(\mathbf{X}_{(n)})$. Tucker decomposition of ranks $S_1, \ldots, S_N$, where $S_n \leq R_n$ seeks a tensor $\mathcal{G} \in \mathbb{R}^{S_1 \times \ldots \times S_N}$ and matrices $\mathbf{A}_{(n)} \in \mathbb{R}^{I_n \times S_n}$, where $n = 1, \ldots, N$ to minimize

$$\|\mathcal{X} - \mathcal{G} \times_1 \mathbf{A}_{(1)} \ldots \times_N \mathbf{A}_{(N)}\|. \tag{7}$$

The decomposition is written briefly as

$$\mathcal{X} \approx [[\mathcal{G}, \mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(n)}]]. \tag{8}$$

If $S_n = R_n$, for all $n = 1, \ldots, N$, we have an exact such decomposition, called the higher-order SVD of $\mathcal{X}$ (HOSVD).

Algorithm 2 Higher-order SVD

Input: $\mathcal{X} \in \mathbb{R}^{I_1, \ldots, I_N}$
  for $n = 1, \ldots, N$ **do**
    $\mathbf{A}_{(n)} \leftarrow R_n$ leading left singular vectors of $\mathbf{X}_(n)$
  end for
    $\mathcal{G} = \mathcal{X} \times_1 \mathbf{A}_{(1)}^\top \ldots \times_N \mathbf{A}_{(N)}^\top$
Output: $\mathcal{G}, \mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(N)}$.

In the case there exists $S_n < R_n$ for some $n = 1, \ldots, N$, taking S_n left singular vectors of $\mathbf{X}_{(n)}$ as in HOSVD does not lead to an optimal solution. We use the Higher-order Orthogonal Iteration (HOOI) approach with HOSVD as an initialization.

Algorithm 3 Higher-order Orthogonal Iteration

Input: $\mathcal{X} \in \mathbb{R}^{I_1, \ldots, I_N}$, $\epsilon = 1e - 10$, maxIter $= 5000$.
  Initialize $\mathbf{A}_{(n)}, n = 1, \ldots, N$ using HOSVD.
  while $\|\mathcal{X} - [[\mathcal{G}, \mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(n)}]]\| > \epsilon$ and iter < maxIter **do**
    for $n = 1, \ldots, N$ **do**
      $\mathcal{Y} \leftarrow \mathcal{X} \times_1 \mathbf{A}_{(1)}^\top \ldots \times_{n-1} \mathbf{A}_{(n-1)}^\top \times_{n+1} \mathbf{A}_{(n+1)}^\top \ldots \times_N \mathbf{A}_{(N)}^\top$
      $\mathbf{A}_{(n)} \leftarrow S_n$ singular vectors of $\mathbf{Y}_{(n)}$
    end for
  end while
    $\mathcal{G} = \mathcal{X} \times_1 \mathbf{A}_{(1)}^\top \ldots \times_N \mathbf{A}_{(N)}^\top$
Output: $\mathcal{G}, \mathbf{A}_{(1)}, \ldots, \mathbf{A}_{(N)}$.

3 Implementation Details

Implementation is organized into three main classes: Tensor, Tensor Builder and Matrix Product. Dependencies are shown in Figure 1.

Testcases can be found in `testcase_generator.m`, including analytically solvable $2 \times 2 \times 2$ tensors to check validity and convergence. Random tensors are also used to check convergence.

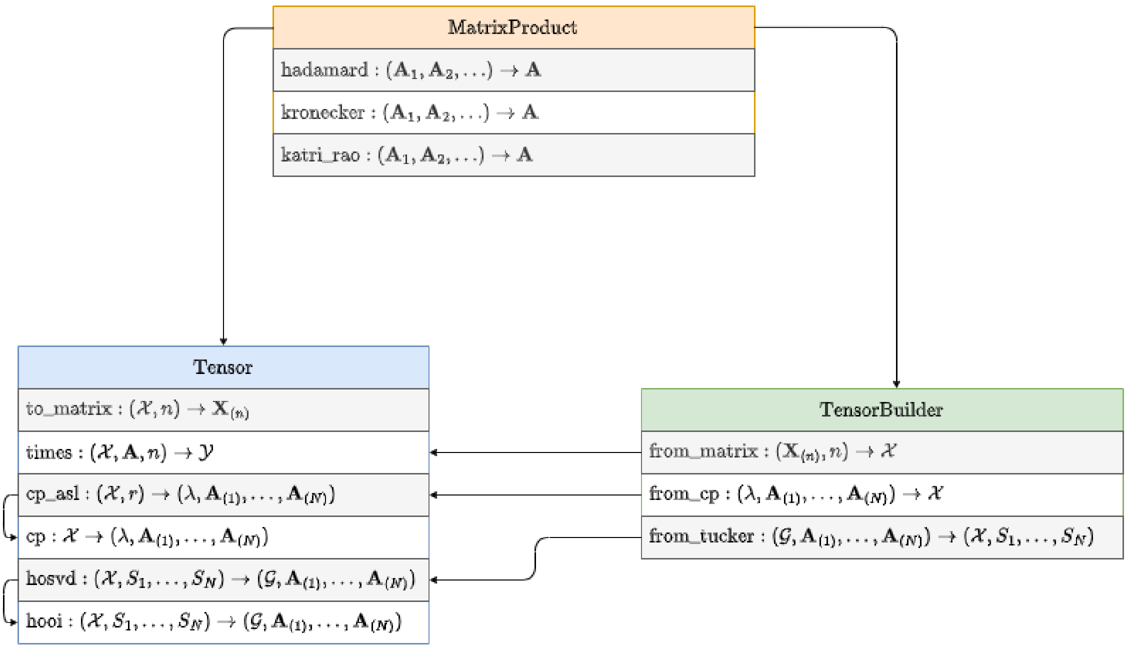

Figure 1: Dependencies between classes and methods

```
>> testcase_generator
----------- TESTCASE 1 ---------------------
CP decomposition...
Tensor of dimensions:      2     2     2

Max possible rank: 4
Approximating by rank-1 tensor...
Reached maximum 5000 iterations.
Final error 8.000000e+00.
------------------------------------------------------------------
Approximating by rank-2 tensor...
Converged after 1 iterations.
Final error 8.283040e-30.
------------------------------------------------------------------
Multilinear ranks not specified, computing max ones...
     1     2     2

     1     1     1

Maximum 5000 iterations reached without convergence.
Final error 8.000000e+00.
----------- TESTCASE 2 ---------------------
CP decomposition...
Tensor of dimensions:      2     2     2

Max possible rank: 4
Approximating by rank-1 tensor...
Reached maximum 5000 iterations.
Final error 3.000000e+00.
------------------------------------------------------------------
Approximating by rank-2 tensor...
```

Figure 2: Test run result

References

[1] Joseph B Kruskal. "Rank, decomposition, and uniqueness for 3-way and N-way arrays". In: *Multiway data analysis*. 1989, pp. 7–18.