Supplemental Materials: Pseudo Code for Nonnegative Tensor Factorizations

Guoxu Zhou and Andrzej Cichocki

I. MATRIZATION OF TENSORS

The required notations and operations for tensors are listed in TALBE I. Note that matricization of tensors plays a central role in the development of tensor decomposition algorithms. As an example, we show the matricizations of a 3rd-order tensor in Fig.[I].

Once the tensor has a special structure, namely Tucker model or CP model (structure), the corresponding matricizations also have special structure, as shown below.

A. Tucker Model

By using the Tucker model, a given tensor $\mathbf{Y} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ is decomposed as

$$\mathbf{Y} = \mathbf{G} \times_1 \mathbf{A}^{(1)} \times_2 \mathbf{A}^{(2)} \cdots \times_N \mathbf{A}^{(N)} = \mathbf{G} \times_{n\in\mathcal{I}_N} \mathbf{A}^{(n)}, \quad (1)$$

$\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times R_n}$ is the mode-$n$ (factor, component) matrix consisting of latent components $a_{r}^{(n)}$ as its columns, $n \in \mathcal{I}_N$, $r \in \mathcal{I}_{R_n}$, and $\mathbf{G} \in \mathbb{R}^{R_1 \times R_2 \cdots \times R_N}$ is the core tensor reflecting interactions between the components in each factor matrix. By using the mode-$n$ matricization of $\mathbf{Y}$, (1) can be rewritten as

$$\mathbf{Y}_{(n)} = \mathbf{A}^{(n)} \mathbf{G}_{(n)} \left[ \bigotimes_{k \in \mathcal{I}_n \setminus \{n\}} \mathbf{A}^{(k)} \right]^T, \quad (2)$$

where $\mathbf{Y}_{(n)}$ and $\mathbf{G}_{(n)}$ are the mode-$n$ matricizations of $\mathbf{Y}$ and $\mathbf{G}$, respectively.

B. CP Model

In CPD, a given tensor $\mathbf{Y}$ can be represented as the sum of rank-1 terms

$$\mathbf{Y} = \sum_{r=1}^{R} \lambda_r \mathbf{a}_{r}^{(1)} \circ \mathbf{a}_{r}^{(2)} \cdots \circ \mathbf{a}_{r}^{(N)}, \quad (3)$$
where \( \odot \) denotes the outer product (see TABLE II). For simplicity, we use \( \mathbf{Y} = [A^{(1)}, A^{(2)}, \ldots, A^{(N)}] \) as a short-hand notation of (3), where \( \lambda_r \) are absorbed into \( A^{(n)} \). By using the mode-\( n \) matricization of \( \mathbf{Y} \), (4) can be written as

\[
\mathbf{Y}^{(n)} = A^{(n)} \left[ \bigodot_{k \in \mathbb{I}_n \setminus \{n\}} A^{(k)} \right]^T.
\]

More details can be found in [1], [2].

**TABLE I: Notations and Operations for Tensors**

<table>
<thead>
<tr>
<th>Notations and Operations</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \mathbf{J} )</td>
<td>A nonnegative matrix</td>
</tr>
<tr>
<td>( \mathbf{A} )</td>
<td>A matrix</td>
</tr>
<tr>
<td>( \mathbf{\theta} )</td>
<td>A vector</td>
</tr>
<tr>
<td>( \mathbf{Y} )</td>
<td>A tensor</td>
</tr>
</tbody>
</table>

**Fibers:** A mode-\( n \) fiber of a tensor is defined by fixing every index but \( n \).

**Matricization:** The mode-\( n \) matricization of \( \mathbf{J} \) yields a \( J_n \)-by-\( \prod_{p \neq n} J_p \) matrix denoted by \( \mathbf{G}(n) \), whose columns consist of all mode-\( n \) fibers of \( \mathbf{J} \).

**Mode-\( n \) product:** The mode-\( n \) product of \( \mathbf{J} \) and \( \mathbf{A} \in \mathbb{R}^{I \times J_n} \) yields a tensor \( \mathbf{Y} = \mathbf{J} \times_n \mathbf{A} \in \mathbb{R}^{J_1 \times \cdots \times J_{n-1} \times I \times J_{n+1} \times \cdots \times J_N} \) whose entries are defined by \( y_{j_1 \cdots j_{n-1} i j_{n+1} \cdots j_N} = \sum_{j_n=1}^{J_n} g_{j_1 j_2 \cdots j_N} a_{i j_n} \). Note that \( \mathbf{Y} = \mathbf{J} \times_n \mathbf{A} \iff \mathbf{Y}^{(n)} = \mathbf{G}(n) \).

**Outer product:** The outer product of \( n \) vectors yields a rank-1 \( n \)-th order tensor. For example, \( \odot \mathbf{b} \odot \mathbf{c} \) yields a 3rd-order tensor \( \mathbf{Y} \) whose elements are defined as \( y_{ijk} = a_{ik} b_{jk} c_k \), where \( \mathbf{a} \), \( \mathbf{b} \), and \( \mathbf{c} \) are vectors.

**Remark 1:** The Khatri-Rao product projection procedure (called KRProj) presented in Algorithm 4 will be frequently used in NTF.

**Remark 2:** In Algorithm 5, the lines 7-8, 11 can be replaced by other nonnegative least squares solvers.

**REFERENCES**


**TABLE II: Notations And Definitions**

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( A ), ( a )</td>
<td>A matrix, the ( r )-th column of matrix ( A ), respectively.</td>
</tr>
<tr>
<td>( \mathbb{I}_N )</td>
<td>The index sets of nonzero integers no larger than ( N ), i.e., ( \mathbb{I}_N = {1, 2, \ldots, N} ).</td>
</tr>
<tr>
<td>( \mathbb{R}_+^{M \times N} )</td>
<td>Set of ( M )-by-( N ) nonnegative matrices. ( \mathbf{A} \in \mathbb{R}_+^{M \times N} \iff \mathbf{A} \succeq 0. )</td>
</tr>
<tr>
<td>( P_r(\mathbf{A}) )</td>
<td></td>
</tr>
<tr>
<td></td>
<td>(</td>
</tr>
<tr>
<td>( \mathbf{Y}, \mathbf{Y}^{(n)} )</td>
<td>A tensor, the mode-( n ) matricization of tensor ( \mathbf{Y} ).</td>
</tr>
<tr>
<td>( \odot, \odot )</td>
<td>Element-wise product and division of matrices (tensors).</td>
</tr>
<tr>
<td>( \odot, \odot )</td>
<td>Kronecker product and Khatri-Rao product (column-wise Kronecker product) of matrices</td>
</tr>
</tbody>
</table>
Generate a 3rd–order tensor

Matlab code:
>>Y=reshape(1:24,[2,3,4])

(a) An example of 3rd-order tensor.

Matricization/unfolding of a 3rd-order tensor

A 2-by-3-by-4 tensor
Matlab code:
>>Y=reshape(1:24,[2,3,4])

• A mode-n fiber (tube) is one column of the mode-n matricization.

Matricization/unfolding

Mode-1 matricization (unfolding)
Matlab code:
>>M=reshape(Y,2,[]);

Mode-2 matricization (unfolding)
Matlab code:
>>M=permute(Y,[2,1,3]);
>>M=reshape(M,3,[]);

Mode-3 matricization (unfolding)
Matlab code:
>>M=permute(Y,[3,1,2]);
>>M=reshape(M,4,[]);

(b) Matricization of a 3rd-order tensor.

Fig. 1: Illustration of matricization operations of tensors.

Algorithm 1: The FastNTF APG Algorithm: Fast Nonnegative Tensor Factorization (NTF) Based on Low-rank Approximation (LRA) and the Accelerated Proximal Gradient (APG) method

Require: Y, J, and any efficient unconstrained CPD algorithm Ψ.
1: $[U(1), U(2), \ldots, U(N)] = Ψ(Y, J)$ is a CPD of Y.
2: Adjust the signs of $U^{(n)}$ and let $A^{(n)} \leftarrow \mathcal{P}_+(U^{(n)})$
3: while not converged do
4:   for $n = 1, 2, \ldots, N$ do
5:     Compute $G = \bigoplus_{p \neq n}(A^{(p)T}A^{(p)})$, $C = U^{(n)} \left(\bigoplus_{p \neq n}(U^{(p)T}A^{(p)})\right)$. $L = \|G\|_F$.
6:     $\alpha_0 = 1$, $k = 1$, and $Z_0 = A_0^{(n)} = A^{(n)}$.
7:     repeat
8:       $A^{(n)}_k = \mathcal{P}_+ \left(Z_{k-1} - \frac{1}{L} \left(C - A^{(n)}_{k-1}G\right)\right)$.
9:       $\alpha_k = \frac{1 + \sqrt{4\alpha_{k-1}^2 + 1}}{2}$.
10:      $Z_k = A^{(n)}_k + \frac{\alpha_{k-1}}{\alpha_k}(A^{(n)}_k - A^{(n)}_{k-1})$.
11:     $k \leftarrow k + 1$
12:   until a stopping criterion is satisfied
13: end for
14: end while {Outer loop}
15: return $Y \approx [A^{(1)}, A^{(2)}, \ldots, A^{(N)}]$ with $A^{(n)} \geq 0$, $\forall n$.

Algorithm 2: Khatri-Rao Product Projection with Nonnegative Constraints: KRProj(H)

Require: $H \in \mathbb{R}^{I \times R}$ and $I_k$, where $\prod_k I_k = I$, $k = 1, 2, \ldots, K$.
1: for $r = 1, 2, \ldots, R$ do
2:     repeat
3:       for $k = 1, 2, \ldots, K$ do
4:         Reshape the $r$th column of $H$ to form the tensor $H^{(r)} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_K}$.
5:         $a_r^{(k)} \leftarrow \mathcal{P}_+ \left(g^{(r)(k)} \prod_{s \neq k} (a^{(s\setminus(k))}_{s\setminus(k)\setminus(r)})\right)$ (Alternatively, $a_r^{(k)}$ can be computed from the left singular vector associated with the leading singular value of $H^{(r)}_{(k)} \in \mathbb{R}^{I_k \times \prod_{p \neq k} I_p}$.)
6:       end for
7:     until a stopping criterion is satisfied.
8: end for
9: return $H \approx \bigodot_k A^{(k)}$, where $A^{(k)} = \begin{bmatrix} a_1^{(k)} & a_2^{(k)} & \ldots & a_R^{(k)} \end{bmatrix} \geq 0$, $k = 1, 2, \ldots, K$.

Algorithm 3: NTF Using Mode Reduction

Require: Y, J, and a NTF algorithm Ψ for 3rd-tensors.
1: Reshape Y to form a 3rd-order tensor $Y^{(3)}$.
2: Let($G^{(1)}, G^{(2)}, G^{(3)}$) $\leftarrow Ψ(Y^{(3)})$.
3: $A^{(n)}$ ($n \in N$) are estimated via efficient Khatri-Rao product projection procedures KRProj($G^{(k)}$), $k = 1, 2, 3$.
4: return $Y \approx [A^{(1)}, A^{(2)}, \ldots, A^{(N)}]$ with $A^{(n)} \geq 0$, $\forall n$. 
Algorithm 4 NTF Based on Unique NMF

Require: $Y$, $J$, and an unique NMF algorithm.

1: Reshape $Y$ to form a matrix $Y^{(2)}$.
2: Run unique NMF to obtain $Y^{(2)} \approx G^{(1)}G^{(2)T}$.
3: $A^{(n)}$ ($n \in \mathcal{N}$) are estimated via efficient Khatri-Rao product projection procedures
   $\text{KRProj}(G^{(k)})$, $k = 1, 2$.
4: return $Y \approx [A^{(1)}, A^{(2)}, \cdots, A^{(N)}]$ with $A^{(n)} \geq 0$, $\forall n$.

Algorithm 5 The LRANTD_MU Algorithm: Fast Nonnegative Tucker Decomposition (NTD) Based on LRA

Require: $Y$, $J_N$.

1: $Y \approx [\mathcal{S}; \tilde{A}^{(1)}, \tilde{A}^{(2)}, \cdots, \tilde{A}^{(N)}]$ by using HOSVD. Initialize $A^{(n)}$, $\mathcal{S}$.
2: while Not converged do
3:   for $n = 1, 2, \ldots, N$ do
4:     $C^{(n)} = A^{(n)T}A^{(n)}$, $\widetilde{C}^{(n)} = \tilde{A}^{(n)T}\tilde{A}^{(n)}$.
5:     $\mathcal{X} = \mathcal{S} \times_{k \in \mathcal{I}_N \setminus \{n\}} C^{(k)} \times_n \tilde{A}^{(n)}$, $\mathcal{B} = \mathcal{S} \times_{k \in \mathcal{I}_N \setminus \{n\}} C^{(k)}$.
6:     repeat
7:       $\mathcal{X} \leftarrow \mathcal{B} \times_n A^{(n)}$.
8:       $A^{(n)} \leftarrow A^{(n)} \odot (\mathcal{X}_n^{(n)}T \mathcal{G}^{(n)}_T) \odot (\mathcal{X}_n \mathcal{G}^{(n)}_T)$.
9:     until a stopping criterion is satisfied.
10: end for
11: Repeat $\mathcal{S} \leftarrow \mathcal{S} \otimes (\mathcal{S} \times_{n \in \mathcal{I}_N} \tilde{C}^{(n)}) \odot (\mathcal{S} \times_{n \in \mathcal{I}_N} C^{(n)})$ until a stopping criterion satisfied.
12: end while
13: return $\tilde{Y} = [\mathcal{S}; A^{(1)}, A^{(2)}, \cdots, A^{(N)}]$. 