INSTITUT NATIONAL DES SCIENCES APPLIQUÉES DE RENNES
Département Génie Mathématique
Promotion 2022

Extrapolation and Sketching for Fast Tensor Decomposition

Kefan Sun

Supervisor: Jérémy Cohen

Rennes, May 27, 2021

Extrapolation and Sketching for Fast Tensor Decomposition

Kefan Sun

Abstract

The CANDECOMP/PARAFAC (CP) decomposition is a leading method for the analysis of multiway data. We study the two variants of Alternating Least Squares algorithm for the CP decomposition (CP-ALS): one with exploration strategy and the other with sketching strategy, which respectively aim at enhance the convergence speed or reduce computation time. A new algorithm is proposed by merging these two algorithms, trying to conserve their advantages. This algorithm can also be adapted for nonnegative decomposition, which has many application fields in real life. In the last section, we propose various numerical tests to asses the performance of the new algorithm.

Code available on Github HER-CPRAND

Contents

Notations

$\mathcal{T}$	A tensor
T_{ijk}	An element of tensor $\mathcal{T} \in \mathbb{R}^{I \times J \times K}$
$\|\|\mathcal{T}\|\|$	The norm Frobenius of tensor $\mathcal{T}$
$T_{(n)}$	The n-mode unfolding of a tensor into a matrix
$\bigotimes$	The Kronecker product of two matrices
$\bigodot$	The Khatri-Rao product of two matrices
$\circ$	The outer product
N	Order (dimensions or modes) of a tensor
R	Rank of a tensor
$\star$	Hadamard (element wise) matrix product
$< \cdot, \cdot >$	Scalar product
†	Pseudo inverse
SNR	Signal-to-noise ratio

1 Introduction

Tensors are generalizations of matrices, they are omnipresent in our real life.

Tensor decomposition is interesting, because it allows to save memory space, or to describe the data in a simpler way and make the data interpretable. But in practice, the data size could be very big and that's why it is important to find a fast tensor decomposition algorithm.

In this document, we will be interested in the CP decomposition, which is a generalization of SVD for matrices. We will firstly study some existing CP decomposition algorithms. The aim of this project is to propose a new fast tensor decomposition algorithm and access its performance.

2 Preliminaries

2.1 Useful operators

Definition 1 *The outer product of vectors $a \in \mathbb{R}^I$ and $b \in \mathbb{R}^J$ results a matrix of size $I \times J$, it is defined by*

$$a \circ b = ab^T = \begin{pmatrix} a_1b_1 & a_1b_2 & \dots & a_1b_J \\ a_2b_1 & a_2b_2 & \dots & a_2b_J \\ \vdots & \vdots & \ddots & \vdots \\ a_Ib_1 & a_Ib_2 & \dots & a_Ib_J \end{pmatrix}.$$

Definition 2 *The Kronecker product of matrices $A \in \mathbb{R}^{I\times J}$ and $B \in \mathbb{R}^{K\times L}$ is denoted by $A \bigotimes B$. The resulting matrix is of size $IK \times JL$, defined by* $A \bigotimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1J}B \\ a_{21}B & a_{22}B & \dots & a_{2J}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{I1}B & a_{I2}B & \dots & a_{IJ}B \end{pmatrix}$.

Definition 3 *The Khatri-Rao product of matrices $A \in \mathbb{R}^{I\times K}$ and $B \in \mathbb{R}^{J\times K}$ is denoted by $A \bigodot B = [a_1 \bigotimes b_1, \dots, a_K \bigotimes b_K]$. This is a columnwise Kronecker product, resulting a matrix of size $IJ \times K$.*

Example 1 $A = \begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{pmatrix}$, $B = \begin{pmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{pmatrix}$, *the Khatri-Rao product of A and B is* $A \bigodot B = \begin{pmatrix} 1 & 8 & 21 \\ 2 & 10 & 24 \\ 3 & 12 & 27 \\ 4 & 20 & 42 \\ 8 & 25 & 48 \\ 12 & 30 & 54 \end{pmatrix}$.

We note that for a given row j of $A \bigodot B$, we can find the corresponding indices of rows of A and B from which it is formed. For example, for $j = 2$, then the tuple $\{$1,2$\}$ corresponds to the row index of A (i.e. 1) and of B(i.e. 2) that gives the j th row of $A \bigodot B$.

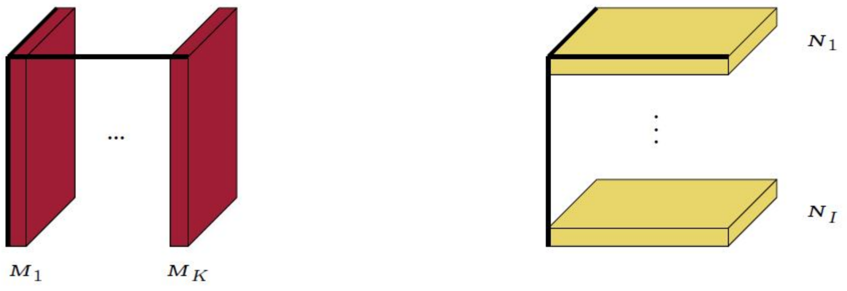
2.2 Tensor norm matricization

To simplify the notation, we consider a 3-mode tensor $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ in this document unless specified otherwise.

Definition 4 *The Frobenius norm of a tensor* $\mathcal{T} \in \mathbb{R}^{I\times J\times K}$ *is* $||\mathcal{T}|| = \sqrt{\sum_{i=1}^{I}\sum_{j=1}^{J}\sum_{k=1}^{K} T_{i,j,k}^2}$.

Definition 5 *The matricization, or unfolding, is the process to sort a tensor into a matrix along a mode.*

There are several different matricizations. The matricization used in Tensorly can be illustrated by:



$$\begin{aligned}\boldsymbol{T}_{(1)} &= [\boldsymbol{M}_1|\ldots|\boldsymbol{M}_K]\\ \boldsymbol{T}_{(2)} &= [\boldsymbol{N}_1|\ldots|\boldsymbol{N}_I]\\ \boldsymbol{T}_{(3)} &= [\boldsymbol{N}_1^T|\ldots|\boldsymbol{N}_I^T]\end{aligned}$$

Fig. 2.1: Three unfoldings of tensor $\mathcal{T}$ - J.Cohen [4]

Example 2 *For a tensor* $\mathcal{T} \in \mathbb{R}^{3\times4\times2}$ *that has* $T_1 = \mathcal{T}_{::1} = \begin{pmatrix} 1 & 4 & 7 & 10\\ 2 & 5 & 8 & 11\\ 3 & 6 & 9 & 12\end{pmatrix}$, $T_2 = \mathcal{T}_{::2} = \begin{pmatrix} 13 & 16 & 19 & 22\\ 14 & 17 & 20 & 23\\ 15 & 18 & 21 & 24\end{pmatrix}$ *as the frontal slices.*

We then have $T_{(1)} = \begin{pmatrix} 1 & 13 & 4 & 16 & 7 & 19 & 10 & 22\\ 2 & 14 & 5 & 17 & 8 & 20 & 11 & 23\\ 3 & 15 & 6 & 18 & 9 & 21 & 12 & 24\end{pmatrix} \in \mathbb{R}^{3\times8}$,

$$T_{(2)} = \begin{pmatrix} 1 & 13 & 2 & 14 & 3 & 15\\ 4 & 16 & 5 & 17 & 6 & 18\\ 7 & 19 & 8 & 20 & 9 & 21\\ 10 & 22 & 11 & 23 & 12 & 24\end{pmatrix} \in \mathbb{R}^{4\times6},$$

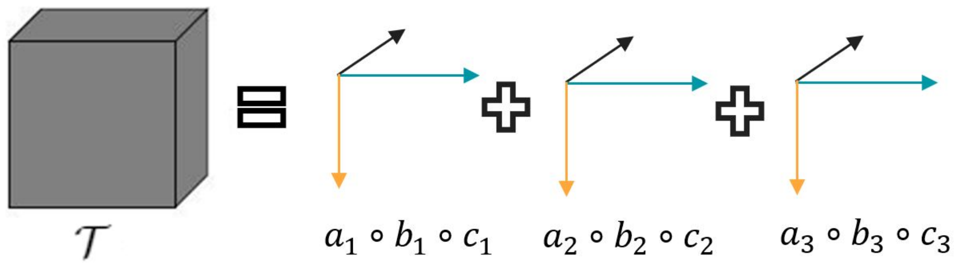
$$T_{(3)} = \begin{pmatrix} 1 & 4 & 7 & 10 & 2 & 5 & 8 & 11 & 3 & 6 & 9 & 12\\ 13 & 16 & 19 & 22 & 14 & 17 & 20 & 23 & 15 & 18 & 21 & 24\end{pmatrix} \in \mathbb{R}^{2\times12}.$$

2.3 Tensor decomposition

Tensors are a generalization of matrices, we would like to generalize the notion of SVD as well. That is exactly what Canonical Polyadic Decomposition (CPD) or the CANDECOMP/PARAFAC Decomposition does[7]. It factorizes a tensor into a sum of R rank-one tensors. That is to say :

$$\mathcal{T} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r \tag{2.1}$$

where $a_r \in \mathbb{R}^I$, $b_r \in \mathbb{R}^J$, $c_r \in \mathbb{R}^K$.



$\mathcal{T}$
$a_1 \circ b_1 \circ c_1$
$a_2 \circ b_2 \circ c_2$
$a_3 \circ b_3 \circ c_3$

Fig. 2.2: Illustration of CPD with $R = 3$

We can gather the vectors of the same mode together to form a **factor matrix**, $A = [a_1 \quad \dots \quad a_R]$, $B = [b_1 \quad \dots \quad b_R]$ and $C = [c_1 \quad \dots \quad c_R]$.

Equation (2.1) can be written by normalizing the columns of factor matrices and expressing the product of the normalization factors as a scalar weight λ_r for each component:

$$\mathcal{T} = \textstyle\sum_{r=1}^{R} \lambda_r a_r \circ b_r \circ c_r$$

This decomposition exists for any tensor if R is large enough, the **rank** of tensor $\mathcal{T}$ is the minimal value of R for which the decomposition is exact.

With column normalization, the CPD is unique if the rank is not too large, up to permutations and signs ambiguities.[4]

A low-rank approximation is interesting because it allows save memory space, and makes it possible to interpret the data with simple patterns. That is the approximate CP Decomposition (aCPD).

The aCPD problem can be written as

$$min_{\hat{\mathcal{T}}\ :\ rank(\hat{\mathcal{T}}) \leq R} ||\mathcal{T} - \hat{\mathcal{T}}|| \tag{2.2}$$

with $\hat{\mathcal{T}}$ defined as in (2.1) for a small R.

But this problem is generally ill-posed for tensors because a minimizer may not exist[4], it is non-convex and NP hard[7].

2.4 Non-negative least squares

The method of least squares is a classical approach to approximate the solution of over-determined systems by minimizing the sum of the squares of the residuals. In some real life applications, such as RBG images and chemometrics, the solution should be non-negative, which means that we need to solve a non-negative least squares (NNLS) problem:

$$min_{x\in\mathbb{R}^d_+}||y - Ax||^2$$

where $y \in \mathbb{R}^m$ and $A \in \mathbb{R}^{m\times d}$.

This problem is convex but not strictly convex unless A is full column rank. Therefore, there does not exist a unique solution x in general. [5]

This problem turns up as subproblems of non-negative matrix/tensor factorization (NM/TF), which are generalizations of NNLS. And we will see in the section 3 that NNLS will be involved in algorithms of aCPD for the non-negative case.

The Hierarchical Alternating Least Square (HALS) is widely used to solve NNLS or NMF problem. For the problem $min_{X\in\mathbb{R}^{d\times n}_+}||Y - AX||^2$ where $Y = [y_1, \dots, y_n] \in \mathbb{R}^{m\times n}$ and $A = [a_1, \dots, a_d] \in \mathbb{R}^{m\times d}$, if x_i denotes the ith row of matrix X, i.e. $X = [x_1, \dots, x_d]^T$, then $AX = \sum_{q=1}^d a_q \circ x_q = \sum_{q=1}^d a_q x_q^T$.

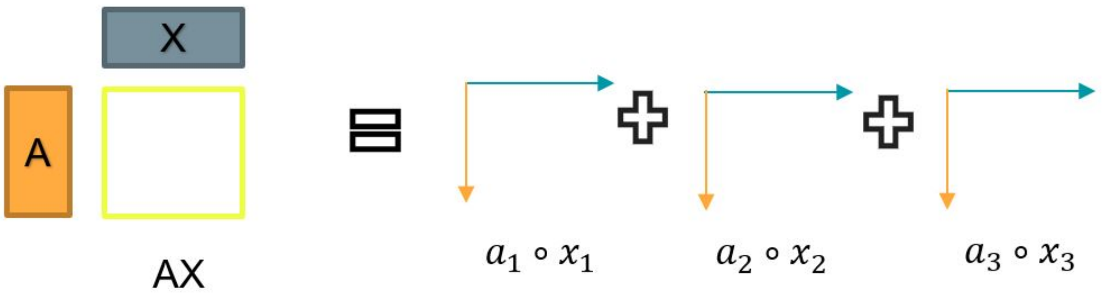


Fig. 2.3: Illustration of HALS with $d = 3$

If we fix all rows x_q but x_j, then the problem becomes $min_{x_j^T\in\mathbb{R}^n_+}||(Y - \sum_{q\neq j} a_q x_q^T) - a_j x_j^T||^2$, the solution $[\frac{a_j^T(Y-\sum_{q\neq j} a_q x_q^T)}{||a_j||^2}]^+$ of this problem can be obtained by extension of the scalar case.

By solving d NNLS problems alternatively, we can approximate the solution X. It has been shown that as long as A has no zero column, the HALS iterates converge towards a minimizer of NNLS [Bertsekas 1995].

Algorithm 1 HALS for NNLS

```
Input: Y, A
1: while Convergence is not met do
2:     for j in [1,...,d] do
3:         Compute $Z = Y - \sum_{q\neq j} a_q \circ x_q$
4:         if $a_j \neq 0$ then
5:             set $x_j = [\frac{a_j^T Z}{||a_j||^2}]^+$
6:         end if
7:     end for
8: end while
```

3 The state of art of aCPD

3.1 CP Alternating Least Squares

Assuming R is fixed, one of the algorithms to compute the aCP decomposition is the Alternating Least Squares (ALS) method.

In fact, (2.1) could be written in matricized form (one per mode)[7]:

$$\begin{aligned}\hat{T}_{(1)} &= A(C \bigodot B)^T \\ \hat{T}_{(2)} &= B(C \bigodot A)^T \\ \hat{T}_{(3)} &= C(B \bigodot A)^T\end{aligned} \tag{3.1}$$

Then the aCPD problem becomes (one per mode):

$$\begin{aligned}&min_{A,B,C}||T_{(1)} - A(C \bigodot B)^T||^2 \\ &min_{A,B,C}||T_{(2)} - B(C \bigodot A)^T||^2 \\ &min_{A,B,C}||T_{(3)} - C(B \bigodot A)^T||^2\end{aligned} \tag{3.2}$$

The idea of ALS is to fix all factor matrices but the ith factor, $\forall i \in \{1, \ldots, N\}$, and to solve the quadratic optimization problem by solving a least-squares problem for the mode i. We repeat the entire procedure until some convergence criterion is satisfied. An illustration is given in Fig.3.1.

For example, suppose that B and C are fixed, the problem becomes

$$min_A||T_{(1)} - A(C \bigodot B)^T||^2 \tag{3.3}$$

It is an over-determined system.

In fact, if we note $Z^{(1)} = C \bigodot B$, or more generally, $Z^{(n)} = A^{(N)} \bigodot \cdots \bigodot A^{(n+1)} \bigodot A^{(n-1)} \bigodot \cdots \bigodot A^{(1)}$ if $\{A^{(n)}\}_{n=1,\ldots,N}$ denotes the factor matrices of a N-order tensor, the normal equation of (3.3) is:

$$T_{(1)}Z^{(1)} = AZ^{(1)T}Z^{(1)}$$

Then the solution is given by the Ordinary Least Squares (OLS). We set $V = Z^{(1)T}Z^{(1)} = C^TC \star B^TB$[7]. $V \in \mathbb{R}^{R\times R}$ is a smaller matrix than the $JK \times R$ matrix $Z^{(1)}$. We set $W = T_{(1)}Z^{(1)} \in \mathbb{R}^{I\times R}$, which is smaller than $I \times JK$.

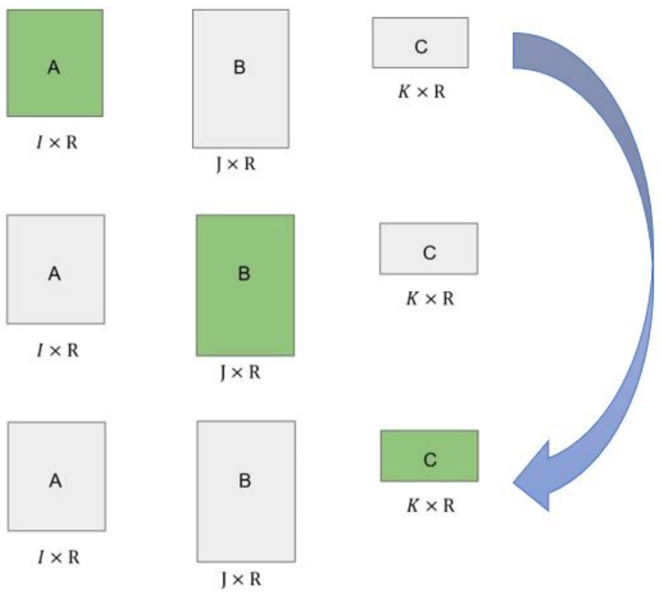


Fig. 3.1: ALS illustration, fixed factors are grey

The CP-ALS algorithm is presented in Algorithm 2. We can initialize the factor matrices either by the leading R left singular vectors of the mode-n unfolding, $T_{(n)}$, or by random matrices.

Algorithm 2 CP-ALS

```
Input: $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, R
Initialize factor matrices $A^{(1)}, \ldots, A^{(N)}$
1: repeat
2:     for $n = 1, \ldots, N$ do
3:         $V \leftarrow A^{(N)T}A^{(N)} \star \cdots \star A^{(n+1)T}A^{(n+1)} \star A^{(n-1)T}A^{(n-1)} \star \cdots \star A^{(1)T}A^{(1)}$
4:         $Z^{(n)} \leftarrow A^{(N)} \bigodot \cdots \bigodot A^{(n+1)} \bigodot A^{(n-1)} \bigodot \cdots \bigodot A^{(1)}$
5:         $W \leftarrow T_{(n)}Z^{(n)}$
6:         Solve $A^{(n)}V = W$ for $A^{(n)}$
7:     end for
8: until some criteria is satisfied
9: return factor matrices $\{A_k^{(n)}\}$
```

One possible termination criterion is to evaluate the cost function (2.2) and terminate the process when the relative residual norm $\frac{||\mathcal{T}-\hat{\mathcal{T}}||}{||\mathcal{T}||}$ is small.

In fact, if $N = 3$, we have

$$\begin{aligned} ||T_{(3)} - C(B \bigodot A)^T||^2 &= ||T_{(3)}||^2 + ||C(B \bigodot A)^T||^2 - 2 < T_{(3)}, C(B \bigodot A)^T > \\ &= ||T_{(3)}||^2 + < C(B \bigodot A)^T, C(B \bigodot A)^T > -2 < T_{(3)}(B \bigodot A), C > \\ &= ||T_{(3)}||^2 + < V, C^T C > -2 < W, C > \end{aligned}$$

to evaluate the cost function. We can use the results V, W of the N-th mode updating step. Then the complexity of cost evaluation is $\mathcal{O}(R^2K)$, which is cheap because R is small. It is much cheaper than $\mathcal{O}(IJKR)$, the complexity of cost evaluation without this tip.

We consider the cost of a single outer iteration of ALS. In line 3, the cost is of $R^2 \sum_{m \neq n} I_m$ to form $A^{(m)T}A^{(m)}$, $m \neq n$, $m \in \{1, \dots, N\}$ and $\mathcal{O}(R^2 N)$ to form V. In line 4, Khatri-Rao product costs $\mathcal{O}(R\Pi_{m\neq n} I_m)$. The line 5 costs $2R\Pi_m I_m$ flops. In line 6, we solve the linear system using Cholesky decomposition, which requires $2R^2 I_n$ flops. So the overall cost of each outer iteration is $\mathcal{O}(NR\Pi_n I_n)$.

Because of OLS, in each step, the cost function can either decrease or stay unchanged, but never increase. The cost is minimized by 0, so it converges, but it is not always guaranteed to converge to a local minima. The approximation quality ultimately depends on initial values for factor matrices.

This algorithm can be easily adapted for Non-negative Tensor Factorization, where factor matrices $\{A^{(n)}\}_{n=1,\dots,N}$ should be non-negative. Since in line 6, we have a least square problem, we could use the HALS algorithm to solve the NNLS problem for NTF. The initial factors should be non-negative as well.

3.2 CP Randomized ALS

Because of high dimensional data and high speed requirements, in order to accelerate the computation speed, researchers have proposed a randomized version of CP-ALS, which extends randomized least squares method to tensors and leverages highly over-determined linear least squares problems.[3]

This algorithm uses the **sketching** technique, which aims to project the data on a lower-dimensional subspace and to solve a smaller LS problem, whose solution is a good approximate of the original problem. For example, consider the problem $min_x ||Ax - b||$ where $A \in \mathbb{R}^{n \times d}$ an over-determined system. We can transform the original problem to a smaller one using some random projection $M \in \mathbb{R}^{S \times n}$ with $S << n$, i.e. $min_x ||MAx - Mb||$, such that the solution of the smaller problem is an approximation of the original problem.

For a tensor $\mathcal{T} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, taking the transpose, the optimization problem (3.3) can be rewritten as $\forall n = \{1, \dots, N\}$:

$$min_{A^{(n)}} ||Z^{(n)} A^{(n)T} - T_{(n)}^T||^2$$

We want to use the sketching technique to solve this LS problem. The sketching technique that we used consists in sampling rows of $Z^{(n)}$, then we need to find the corresponding rows of $T_{(n)}^T$. Here we consider a uniform sampling with replacement, which is the simplest sampling method. Let $\mathcal{S}$ denote the uniform samples from $\{1, \dots, \Pi_{m\neq n} I_m\}$ and $|\mathcal{S}|$ the sample size, the sampling operation can be expressed as the application of a selection matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{S}| \times \Pi_{m\neq n} I_m}$, where the rows of $\mathbf{S}$ are rows of the $\Pi_{m\neq n} I_m \times \Pi_{m\neq n} I_m$ identity matrix.

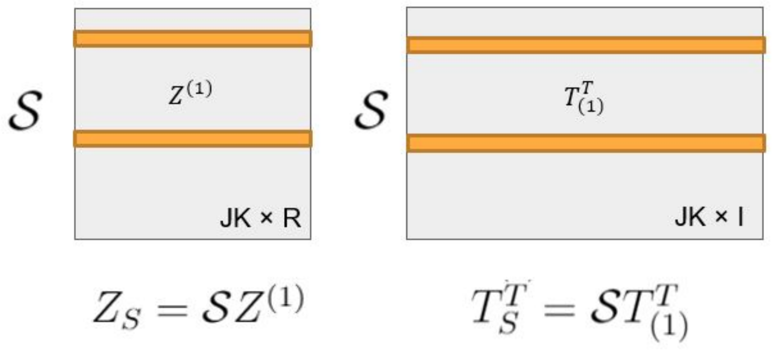


Fig. 3.2: CPRAND - sketching illustration with n=1

If we want to compute the sketch matrix $\mathbf{S}Z^{(n)}$ without explicitly forming $Z^{(n)}$, it can be done with the help of Algorithm 3, where **idxs** is the set of tuples $\{i_1^{(j)}, \dots, i_{n-1}^{(j)}, i_{n+1}^{(j)}, \dots, i_N^{(j)}\}$ for the selected row $j \in \mathcal{S}$ of $Z^{(n)}$. **idxs** can be easily found because of the property of Khatri-Rao product explained in section 2.1.

Algorithm 3 Sampled Khatri-Rao Product

```
Input: S, $A^{(N)}, \dots, A^{(n+1)}, A^{(n-1)}, \dots, A^{(1)}$
Retrieve **idxs** from S
$Z_S \leftarrow 1 \in \mathbb{R}^{|\mathcal{S}| \times R}$
1: **for** $m = 1, \dots, n-1, n+1, \dots, N$ **do**
2:     $A_S^{(m)} \leftarrow A^{(m)}(\mathbf{idxs}(:, m), :)$
3:     $Z_S \leftarrow Z_S \star A_s^{(m)}$
4: **end for**
return $Z_S$
```

In the same way that we want to avoid forming $Z^{(n)}$ explicitly, we also want to avoid forming $T_{(n)}$. In fact, $\mathbf{S}T_{(n)}^T$ can be formed using **idxs** of Sampled Khatri-Rao product. $\forall j \in \mathcal{S}$ and **idxs** the set of tuples corresponding to j, the fiber $T_{i_1^{(j)}, \dots, i_{n-1}^{(j)}, :, i_{n+1}^{(j)}, \dots, i_N^{(j)}}$ corresponds to the j th column of $T_{(n)}$ (so j th row of $T_{(n)}^T$), for the matricization proposed in [7]. The randomized version of CP-ALS is named CPRAND (see Algorithm 4).

Algorithm 4 CPRAND

```
Input: $\mathcal{T} \in \mathbb{R}^{I_1 \times \dots \times I_N}$, R, $|\mathcal{S}|$
Initialize factor matrices $A^{(1)}, \dots, A^{(N)}$
1: **repeat**
2:     **for** $n = 1, \dots, N$ **do**
3:         Define sampling operator $S \in \mathbb{R}^{|\mathcal{S}| \times \Pi_{m \neq n} I_m}$
4:         $Z_S \leftarrow SKR(S, A^{(1)}, \dots, A^{(n-1)}, A^{(n+1)}, \dots, A^{(N)})$
5:         $X_S^T \leftarrow SX_{(n)}^T$
6:         $A^{(n)} \leftarrow argmin_A ||Z_S A^T - X_S^T||$
7:     **end for**
8: **until** some criteria is satisfied
return factor matrices $\{A_k^{(n)}\}$
```

A sample size of $S = 10Rlog(R)$ has been proven sufficient, provided that the data is incoherent [3].

In terms of termination criterion, we could use the same criterion as for ALS. The robustness of CPRAND is the low cost of computation, to keep the per-iteration complexity low, the error should also be computed on smaller-scale, sketched data, e.g. compute the norm of residual within the sample used for update step. Unfortunately, the variance of this value is very high due to the small size of samples. That's why the authors of [3] proposed an alternative.

This alternative consists in sampling $\hat{P}$ indices of $\mathcal{T}$, with $IJK >> \hat{P} >> |\mathcal{S}|$, compute the mean residual norm on this sample, then the total residual estimation can be obtained by multiplying the estimate of mean residual norm by the total number of indices $P = \Pi_n I_n$. Let $\hat{I}$ be the random subset of $\hat{P}$ indices, and $\mathbf{i} = (i_1, \dots, i_N)$ be a multi-index, then $||\mathcal{T} - \hat{\mathcal{T}}|| \approx (P\hat{\mu})^{1/2}$, where $\hat{\mu} = mean\{(T_\mathbf{i} - \hat{T}_\mathbf{i})^2 \mid \mathbf{i} \in \hat{I}\}$. A confidence of 98% is maintained when the number of samples $\hat{P} \geq 372\mu_{max}$ where μ_{max} is the maximum allowable value of residual for each index. We usually assume $\mu_{max} = 1$[3]. The complexity of cost evaluation is $\mathcal{O}(\hat{P}RN)$.

Given that the approximate error are not guaranteed to decrease at each iteration because of the randomization, in practice, the simplest strategy is to store the lowest relative error achieved so far, and terminate when a particular number of iterations have elapsed without any reduction in this minimum. This number of allowable iterations is denoted by err_it_max.

We consider the cost of an outer iteration. In line 3, we use $\mathcal{O}(|\mathcal{S}|N)$ operations to generate the sampling operator $\mathcal{S}$. In line 4, the cost of algorithm 2 is $|\mathcal{S}|R(N-1)$ flops. In line 5, sampling $|\mathcal{S}|$ fibers from $\mathcal{T}$ to form $T_\mathcal{S}$ requires negligible computation time. In line 6, we solve the sampled linear system by $2|\mathcal{S}|R^2$ flops using QR. The cost of computing $Q^T T_\mathcal{S}^T$ is $2|\mathcal{S}|RI_n$ operations, and the triangular solve is $R^2 I_n$ operations. Assuming $I_n > R$ and $|\mathcal{S}| > R$, the leading cost is $2|\mathcal{S}|RI_n$ operations. So the overall cost is $\mathcal{O}(|\mathcal{S}|R\sum_n I_n)$.

As for ALS, this algorithm can be easily adapted for non-negative decomposition, we can use the HALS algorithm to solve the NNLS problem in line 6 for NTF. The initial factors should also be non-negative.

3.3 Heuristic Extrapolation Restart ALS

ALS is known to converge slowly when the factors $A^{(j)}$ are ill-conditioned. A procedure of Heuristic Extrapolation and Restart (HER) was introduced in [2]. This procedure mimics Nesterov's extrapolation by introducing pairing variables $\hat{A}^{(j)}$. It consists in extrapolating the factor estimates between each block update.

The Nesterov's extrapolation can accelerate the convergence rate by exploring a new direction and make a correction to the jump. In HER-ALS, we have two sequences, the factors $A^{(j)}$ and the extrapolated sequence (our pairing variables) $\hat{A}^{(j)}$. $\hat{A}^{(j)}$ represents the new direction that is obtained by a jump based on the previous update of $A^{(j)}$. The update of each factor is based on the extrapolated sequence $\hat{A}^{(j)}$ (so the new direction).

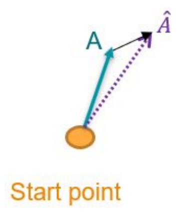


Fig. 3.3: HER illustration

Then we compare the cost F_k with the previous cost F_{k-1}. When F increases, the new extrapolated direction is a worse one, we assume it is caused by an over-sized step, so $\{\hat{A}_k^{(i)}\}_{i=1,\dots,N}$ is abandoned by performing a restart and decreasing the step. Otherwise, the new extrapolated direction is a better one, we assume the step can safely be increased, so the $\{\hat{A}_k^{(i)}\}_{i=1,\dots,N}$ is keeped as factor matrices and we increase the step.

When updating the ith factor in the ALS algorithm at the kth iteration, the update is modified as :

$$A_k^{(i)} = g(\mathcal{T}, \hat{A}_k^{(j<i)}, \hat{A}_{k-1}^{(j>i)}) := \hat{W}\hat{V}^\dagger \tag{3.4}$$

with $\hat{V} = \hat{A}_{k-1}^{(N)T}\hat{A}_{k-1}^{(N)} \star \cdots \star \hat{A}_{k-1}^{(i+1)T}\hat{A}_{k-1}^{(i+1)} \star \hat{A}_k^{(i-1)T}\hat{A}_k^{(i-1)} \star \cdots \star \hat{A}_k^{(1)T}\hat{A}_k^{(1)}$ and $\hat{W} = T_{(i)}\hat{Z}^{(i)}$ $= T_{(i)}(\hat{A}_{k-1}^{(N)} \bigodot \cdots \bigodot \hat{A}_{k-1}^{(i+1)} \bigodot \hat{A}_k^{(i-1)} \bigodot \cdots \bigodot \hat{A}_k^{(1)})$.

The extrapolation is :

$$\hat{A}_k^{(i)} = A_k^{(i)} + \beta_{k-1}(A_k^{(i)} - A_{k-1}^{(i)}) \tag{3.5}$$

where β_k is updated heuristically (see Algorithm 5) using three parameters $(\eta, \bar{\gamma}, \gamma)$, where η is the decay rate of β, γ the growth rate of β and $\bar{\gamma}$ the growth rate of $\bar{\beta}$ (the upper bound of β).

Algorithm 5 HER-ALS

Input: $\mathcal{T} \in \mathbb{R}^{I_1\times\cdots\times I_N}$, R

Initialization : Choose $\beta_0 \in (0,1)$, $\eta \geq \gamma \geq \gamma \geq 1$, factor matrices $A_0^{(1)}, \dots, A_0^{(N)}$ and $\hat{A}_0^{(1)}, \dots, \hat{A}_0^{(N)}$. Set $\bar{\beta}_0 = 1$ and $k = 1$.

```
1:  repeat
2:      for $i = 1, \dots, N$ do
3:          Update: $A_k^{(i)} = g(\mathcal{T}, \hat{A}_k^{(j<i)}, \hat{A}_{k-1}^{(j>i)})$
4:          Extrapolate: $\hat{A}_k^{(i)} = A_k^{(i)} + \beta_{k-1}(A_k^{(i)} - A_{k-1}^{(i)})$
5:      end for
6:      Compute $\hat{F}_k = F(A_k^{(N)}; \hat{A}_k^{(l\neq N)})$.
7:      if $\hat{F}_k > \hat{F}_{k-1}$ for $k \geq 2$ then
8:          Set $\hat{A}_k^{(i)} = A_k^{(i)}$ for $i = 1, \dots, N$
9:          Set $\bar{\beta}_k = \beta_{k-1}$, $\beta_k = \beta_{k-1}/\eta$
10:     else
11:         Set $A_k^{(i)} = \hat{A}_k^{(i)}$ for $i = 1, \dots, N$
12:         Set $\bar{\beta}_k = min\{1, \bar{\beta}_{k-1}\bar{\gamma}\}$, $\beta_k = min\{\bar{\beta}_{k-1}, \gamma\beta_{k-1}\}$
13:     end if
14:     Set k=k+1
15: until some criteria is satisfied
return factor matrices $\{A_k^{(i)}\}$
```

In line 6, the restart criterion is $\hat{F}_k = F(A_k^{(N)}; \hat{A}_k^{(l\neq N)})$, which is the cost evaluated at the pairing variable for the first N-1 modes and the factors sequence at the last mode. Such a criterion is chosen because in the same way as for ALS algorithm, we can evaluate the criterion using values computed for the update step without doing extra computation. The cost $\hat{F}$ asymptotically converges to F if the two sequences converge to the same limit point, e.g. the true factors. If $N = 3$, we have:

$$\hat{F} = ||T_{(3)}||^2 + < \hat{V}, C^TC > -2 < \hat{W}, C > .$$

The default parameters are $[\beta_0, \gamma, \bar{\gamma}, \eta] = [0.5, 1.05, 1.01, 1.5]$. [2]

Note that we do not normalize factors $A^{(j)}$ or pairing variables $\hat{A}^{(j)}$ at each iteration. In fact, it is difficult to manage the normalization of two sequences that are related to each other. If we do so, we would change the jump step for extrapolation (line 4), and numerically we would find the error oscillates from one iteration to another.

For an outer iteration, the additional cost of HER $\mathcal{O}(R\sum_n I_n)$ comes from line 4, and it can be neglected compared with the larger cost of ALS $\mathcal{O}(NR\Pi_n I_n)$.

To adapt this algorithm for the non-negative case, one needs to use the HALS algorithm to solve the NNLS problem in line 3. The extrapolated sequence could be non-negative as well, i.e. replace the line 4 by $\hat{A}_k^{(i)} = \max(0, A_k^{(i)} + \beta_{k-1}(A_k^{(i)} - A_{k-1}^{(i)}))$. The initial factors should be non-negative too.

4 HER Randomized ALS

We would like to propose a new algorithm that merges HER-ALS and CPRAND and conserves their advantages: HER-ALS enhances convergence speed by adding an exploration direction and CPRAND does less computation by sampling the tensor.

We refer the new algorithm as HER-CPRAND. In HER-ALS, instead of computing exactly $\hat{V}$ and $\hat{W}$ of (3.4) to solve the LS problem, we apply CPRAND's sketching strategy to pairing variables $\{\hat{A}^{(n)}\}$. For the mode n, we can obtain $\hat{Z}_S = SKR(S, \hat{A}^{(1)}, \ldots, \hat{A}^{(n-1)}, \hat{A}^{(n+1)}, \ldots, \hat{A}^{(N)})$, then $\hat{V}_k^S = \hat{Z}_S^T \hat{Z}_S$ and $\hat{W}_k^S = T_S \hat{Z}_S$, where T_S can be formed with the indices used in SKR.

We can then calculate the factor matrices (line 6) with the randomized $\hat{V}_S$ and $\hat{W}_S$, and extrapolate as in HER-ALS (line 7).

Algorithm 6 HER-CPRAND

```
Input: $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, R, $|\mathcal{S}|$
Initialization: Choose $\beta_0 \in (0,1)$, $\eta \geq \gamma \geq \bar{\gamma} > 1$, and 2 sets of initial factor matrices $(A_0^{(1)}, \ldots, A_0^{(N)})$ and $(\hat{A}_0^{(1)}, \ldots, \hat{A}_0^{(N)})$. Set $\bar{\beta}_0 = 1$ and $k = 1$.
 1: repeat
 2:     for $n = 1, \ldots, N$ do
 3:         Define sampling operator $\mathcal{S} \in \mathbf{R}^{|\mathcal{S}| \times \prod_{m \neq n} I_m}$
 4:         $\hat{Z}_S \leftarrow SKR(\mathcal{S}, \hat{A}_k^{(1)}, \ldots, \hat{A}_k^{(n-1)}, \hat{A}_{k-1}^{(n+1)}, \ldots, \hat{A}_{k-1}^{(N)})$
 5:         $T_S^T \leftarrow \mathcal{S} T_n^T$
 6:         Update : $A_k^{(n)} \leftarrow argmin_A || \hat{Z}_S A^T - T_S^T ||$
 7:         Extrapolate : $\hat{A}_k^{(n)} = A_k^{(n)} + \beta_{k-1}(A_k^{(n)} - A_{k-1}^{(n)})$
 8:     end for
 9:     Compute $F_k = F(A_k^{(1)}, \ldots, A_k^{(N)})$.
10:     if $F_k > F_{k-1}$ for $k \geq 2$ then
11:         Set $\hat{A}_k^{(i)} = A_k^{(i)}$ for $i = 1, \ldots, N$
12:         Set $\bar{\beta}_k = \beta_{k-1}$, $\beta_k = \beta_{k-1}/\eta$
13:     else
14:         Set $A_k^{(i)} = \hat{A}_k^{(i)}$ for $i = 1, \ldots, N$
15:         Set $\bar{\beta}_k = min\{1, \bar{\beta}_{k-1}\bar{\gamma}\}$, $\beta_k = min\{\bar{\beta}_{k-1}, \gamma\beta_{k-1}\}$
16:     end if
17:     Set k=k+1
18: until some criteria is satisfied
return $\lambda$, factor matrices $\{A_k^{(n)}\}$
```

Note that as explained in the Section 3.3, because of the HER procedure, we do not normalize our factors at each iteration.

It seems simple to merge these two algorithms. But the tricky part is the error evaluation (line 9 and 10). In fact, for HER procedure, we need to be able to compare the error (or cost) of two successive iterations. And the error evaluation should be cheap and accurate to keep the advantage of CPRAND.

One possible idea is to continue to benefit from the fast error estimation of HER-ALS. To make the error estimations comparable, we need actually to do one more error computation at each iteration: for the k-th iteration, if $N = 3$, let $\mathcal{S}$ be the last sampling operator generated for $n = 3$, the cost at this iteration with the samples $\mathcal{S}$ is denoted by $\hat{F}_k^S$. We have

$$(\hat{F}_k^S)^2 = ||T_S||^2 + < \hat{V}_k^S, A_k^{(3)T} A_k^{(3)} > -2 < \hat{W}_k^S, A_k^{(3)} > .$$

From $\hat{A}_{k-1}^{(1)}$, $\hat{A}_{k-1}^{(2)}$ and $A_{k-1}^{(3)}$, we need to compute

$$(\hat{F}_{k-1}^S)^2 = ||T_S||^2 + < \hat{V}_{k-1}^S, A_{k-1}^{(3)\;T} A_{k-1}^{(3)} > -2 < \hat{W}_{k-1}^S, A_{k-1}^{(3)} >$$

with the same sampling operator. Then we compare $\hat{F}_k^S$ with $\hat{F}_{k-1}^S$ to decide whether we restart or not. But the problem is that the factors at the k th iteration $A_k^{(n)}$ are optimal for the sample $\mathcal{S}$, that's why we usually have $\hat{F}_{k-1}^S \geq \hat{F}_k^S$, we will not restart even though it should occur. So we will not detect immediately that we are in a worse direction.

What's more, because of the small number of samples used for sketching, the variance of the error estimation is very high as stated in the Section 3.2.

Another idea is to use the error estimation proposed in the section 3.2, and the sample used for two successive iterations should be the same. To do so, we could either use one same sample of error estimation for all iterations, or use different sample at each iteration. But the later requires one more error computation at each iteration: for the k th iteration, let $\mathcal{S}_k^{err}$ be the sample used for error estimation F_k^S at this iteration, we need to evaluate F_{k-1}^S (with $A_{k-1}^{(1)}, \dots, A_{k-1}^{(N)}$) on the same sample $\mathcal{S}_k^{err}$.

One would like to reduce the random effect to have a more accurate error comparison. For example, we can compare the estimation at kth iteration F_k^S with the mean value of the last 10 estimations $\bar{F}_{k-1} = mean\{F_i^S\ |i = k-1, \dots, k-10\}$.

We have four combinations possible:

- Use one same sample for all iterations without taking mean value
- Use different samples for each iteration without taking mean value
- Use one same sample for all iterations, take mean value on 10 last estimations
- Use different samples for each iteration, take mean value on 10 last estimations

The fact that we use different sample for each iteration doesn't improve the accuracy of estimation or reduce the variance. The only advantage is that it allows us to abandon bad $\mathcal{S}^{err}$, and we don't keep the bad error samples during all iterations of the algorithm. But it takes more time because of the double computation. So we abandon the second and the fourth option. Using the mean filter does stabilize the error estimation (i.e. the variance is smaller). Because the factors at each iteration are computed with different samples $\mathcal{S}$, by taking the mean value, we can stabilize this random effect without introducing other random source (e.g. different samples $\mathcal{S}^{err}$). That's why we have implemented the third option (see Algorithm 7).

Algorithm 7 Cheap HER-CPRAND

```
Input: $\mathcal{T} \in \mathbb{R}^{I_1 \times \cdots \times I_N}$, R, $|\mathcal{S}|$, $\hat{P}$
Initialization: Choose $\beta_0 \in (0,1)$, $\eta \geq \gamma \geq \bar{\gamma} > 1$, and 2 sets of initial factor matrices $(A_0^{(1)}, \ldots, A_0^{(N)})$ and $(\hat{A}_0^{(1)}, \ldots, \hat{A}_0^{(N)})$. Set $\bar{\beta}_0 = 1$ and $k = 1$. Define sampling operator $\mathcal{S}^{err} \in \mathbf{R}^{\hat{P} \times \prod_{m=1}^{N} I_m}$.
1:  repeat
2:      for $n = 1, \ldots, N$ do
3:          Define sampling operator $\mathcal{S} \in \mathbf{R}^{|\mathcal{S}| \times \prod_{m \neq n} I_m}$
4:          $\hat{Z}_S \leftarrow SKR(\mathcal{S}, \hat{A}_k^{(1)}, \ldots, \hat{A}_k^{(n-1)}, \hat{A}_{k-1}^{(n+1)}, \ldots, \hat{A}_{k-1}^{(N)})$
5:          $T_S^T \leftarrow \mathcal{S} T_n^T$
6:          Update : $A_k^{(n)} \leftarrow argmin_A ||\hat{Z}_S A^T - T_S^T||$
7:          Extrapolate : $\hat{A}_k^{(n)} = A_k^{(n)} + \beta_{k-1}(A_k^{(n)} - A_{k-1}^{(n)})$
8:      end for
9:      Compute $F_k^S = F(A_k^{(1)}, \ldots, A_k^{(N)}, \mathcal{S}^{err})$
10:     if $F_k^S > F_{k-1}^S$ for $k \geq 2$ then
11:         Set $\hat{A}_k^{(i)} = A_k^{(i)}$ for $i = 1, \ldots, N$
12:         Set $\bar{\beta}_k = \beta_{k-1}$, $\beta_k = \beta_{k-1}/\eta$
13:     else
14:         Set $A_k^{(i)} = \hat{A}_k^{(i)}$ for $i = 1, \ldots, N$
15:         Set $\bar{\beta}_k = min\{1, \bar{\beta}_{k-1}\bar{\gamma}\}$, $\beta_k = min\{\bar{\beta}_{k-1}, \gamma\beta_{k-1}\}$
16:     end if
17:     Set $F_k^S = mean\{F_i^S | i = k, \ldots, k-9\}$
18:     Set k=k+1
19: until some criteria is satisfied
return $\lambda$, factor matrices $\{A_k^{(n)}\}$
```

Without considering the line 9, the cost of an outer iteration of HER-CPRAND is $\mathcal{O}(SR\sum_n I_n)$, like CPRAND, because the extrapolation costs $\mathcal{O}(R\sum_n I_n)$, which can be neglected. The cost of error estimation is $\mathcal{O}(\hat{P}RN)$, so the overall cost is $\mathcal{O}(SR\sum_n I_n + \hat{P}RN)$.

This algorithm can be easily adapted in the non-negative case, in line 6 we can use the HALS algorithm to solve the NNLS problem, and the extrapolated sequence should be non-negative as well, i.e. replace the line 7 by $\hat{A}_k^{(i)} = \max(0, A_k^{(i)} + \beta_{k-1}(A_k^{(i)} - A_{k-1}^{(i)}))$. The initial factors should be non-negative too.

5 Experiments

We use two metrics to measure the quality of aCPD.

The fit, which measures how well the data is recovered, is defined as $F = 1 - \frac{f}{||\mathcal{T}||}$, with f the data fitting error (the value of the objective function $f := ||\mathcal{T} - \hat{\mathcal{T}}||$).

The score, which measures how well the ground truth factors are recovered, is $score(\mathcal{T}, \hat{\mathcal{T}}) = \frac{1}{R}\sum_{r=1}^{R}(\frac{a_r^T \hat{a}_r}{||a_r||||\hat{a}_r||} \times \frac{b_r^T \hat{b}_r}{||b_r||||\hat{b}_r||} \times \frac{c_r^T \hat{c}_r}{||c_r||||\hat{c}_r||})$, where $\mathcal{T} = \sum_{r=1}^{R} a_r \circ b_r \circ c_r$, and $\hat{\mathcal{T}} = \sum_{r=1}^{R} \hat{a}_r \circ \hat{b}_r \circ \hat{c}_r$. The columns of estimated factor matrices $\hat{A}$, $\hat{B}$, $\hat{C}$ are arranged in the same order as in the true factors A, B and C with the help of the Hungarian algorithm which solves the linear assignment problem. In fact, at convergence, the estimated factors converge to the true factors, up to permutations and signs ambiguities. If the true factors and the estimated factors have unit-norm columns, then $|(A^T\hat{A})_{i,j}| = 1$ if the j-th column of $\hat{A}$ corresponds to the i-th column of A. One can apply the Hungarian algorithm to find a maximal cost assignment in order to rearrange the columns of $\hat{A}$.

5.1 Parameter search

Firstly, we need to find the optimal extrapolation parameters for HER-CPRAND.

We run HER-CPRAND with different parameters on 10 randomly generated rank-10 tensor of size $100 \times 100 \times 100$. To do so, we generate randomly factor matrices from standard normal distribution, and we scale the condition number of each factor matrix to be 10, i.e. scale the singular values to be $1, \dots, 10$, to introduce some correlation and to mimic a real-life scenario. Then we add a noise term $\mathcal{N}$ to tensors. The noise is drawn from standard normal distribution, so we have $\mathcal{T} = \mathcal{T}_{true} + \theta\frac{||\mathcal{T}_{true}||}{||\mathcal{N}||}\mathcal{N}$, with $\theta = 0.1$ ($SNR \approx 20dB$) the noise level.

For each tensor, we use 5 random initialization of factors from uniform distribution on $[0, 1)$. The sample size is $|\mathcal{S}| = 10Rlog(R)$. The stopping criterion is the exact relative residual norm. We display the exact data fitting error f along with number of iterations.

For the extrapolation step $\beta \in [0, 1]$, we test 3 values, 0.1, 0.5 and 0.9. The blue curves decrease faster than the others, so we take $\beta = 0.1$.

For the decay rate $\eta \in [\gamma, \infty)$, there is no significant difference, we take $\eta = 3$.

For the growth rate $\gamma \in [\bar{\gamma}, \eta]$ and $\bar{\gamma} \in [1, \gamma]$, the green curves decrease more slowly than the others. The blue curves are slightly better. We take $[\gamma, \bar{\gamma}] = [1.01, 1.005]$.

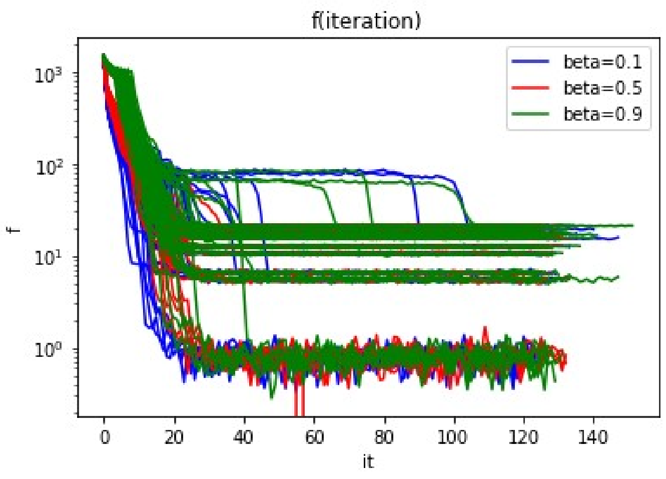


Fig. 5.1: β

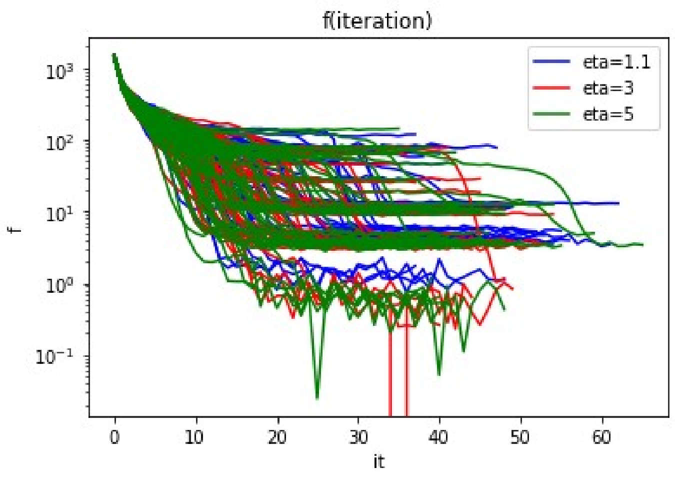


Fig. 5.2: η

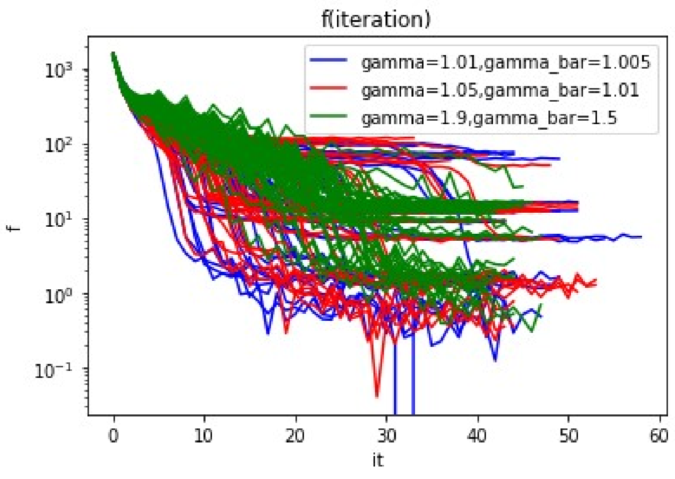


Fig. 5.3: γ

5.2 Speed up

We want to investigate the speed up factor of HER-CPRAND versus ALS, HER-ALS and CPRAND for different data size N and different sample size $|\mathcal{S}|$.

For a given data size N and sample size $|\mathcal{S}|$, we generate randomly 5 noised ($\theta = 0.1$) rank-10 tensors of size $N \times N \times N$ and scale the condition number of true factors to 10. We use one same random initial factors for the 5 tensors for all algorithms. The algorithms terminate if the relative residual norm goes below $tol = 1.1 * \theta$ (it can't be less than the noise level and we tolerate 10% error). The data fitting error is estimated using $\hat{P} = 400$ samples for (HER-)CPRAND. The speed up factor for a generic algorithm A is defined as $\bar{T}_A/\bar{T}_{HER-CPRAND}$, where $\bar{T}_A$ denotes the mean time of algorithm A for the 5 tensors.

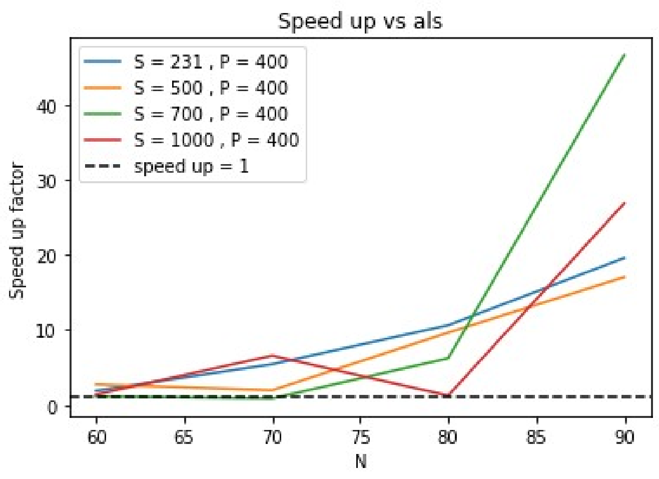


Fig. 5.4: vs ALS

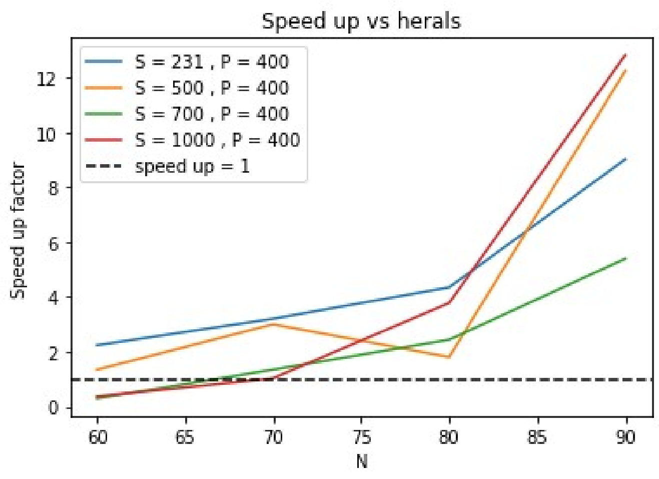


Fig. 5.5: vs HER-ALS

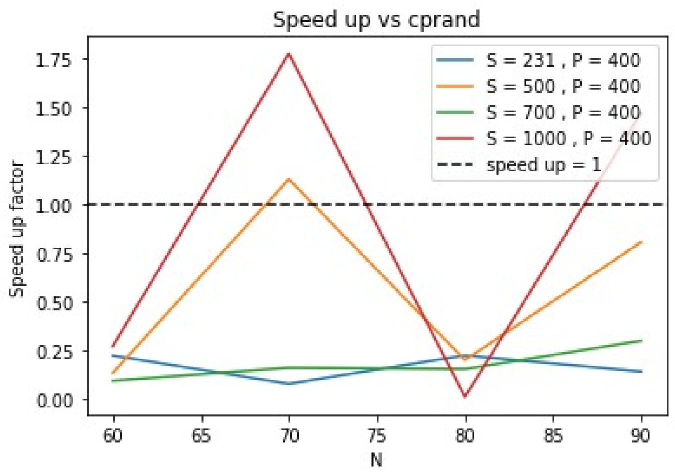


Fig. 5.6: vs CPRAND

The bigger N is, the bigger the speed up factor is for (HER-)ALS. The speed up factor is bigger for ALS than for HER-ALS with the same trend, which shows the efficacy of HER procedure.

We note that the blue curves ($S = 231$, the smallest sample size) are efficient for small data e.g. $N \leq 80$, it becomes less efficient for big data. Even though we do less computation at each iteration, it demands more iterations, and finally it takes more time e.g. for $N = 90$. For big data, the sample size $|\mathcal{S}|$ should be bigger not only for the accuracy but also for the speed up.

From Fig.5.6 we can see that we don't always have a mean computation time improvement with respect to CPRAND. This figure shows us the improvement of HER procedure, and we can see that the speed up factor is very random, because we use random initial factors and we cannot control the distance from our initial factors to true factors.

Now we do the same thing for the non-negative case.

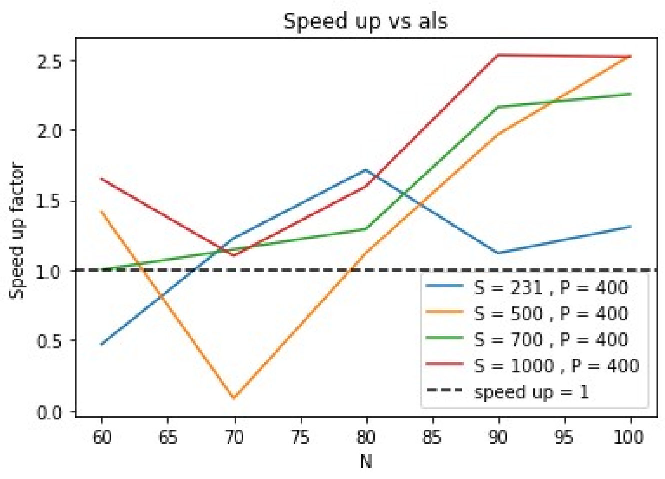


Fig. 5.7: NN vs ALS

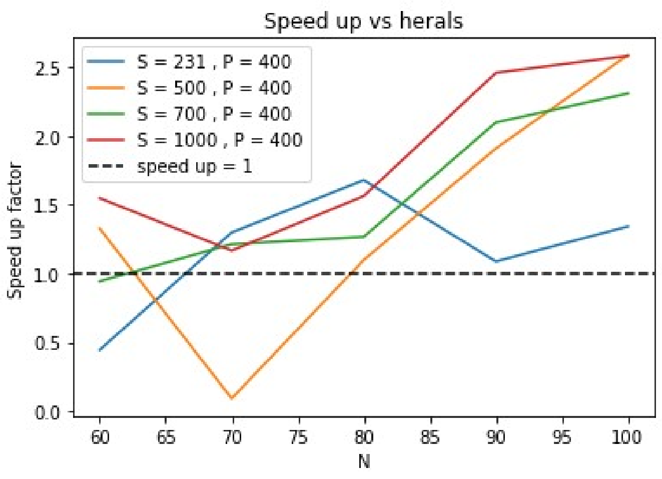


Fig. 5.8: NN vs HER-ALS

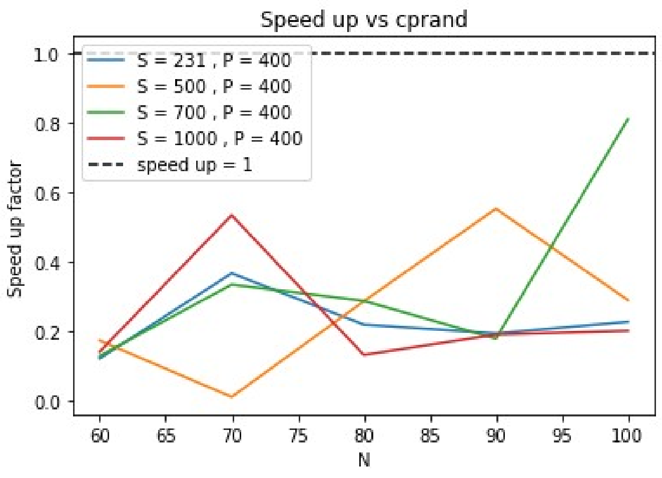


Fig. 5.9: NN vs CPRAND

Here the speed up factor is smaller than the previous case, and there is no significant difference between HER-ALS and ALS. We do not have an improvement compared to CPRAND.

5.3 Synthetic data

Then we compare the four algorithms with respect to the quality of aCPD, the number of iterations, time and the percentage of restart on synthetic data.

We create randomly $nb\_rand = 10$ 3-order noised ($\theta = 0.3$, $SNR \approx 10dB$) tensors of size $100 \times 100 \times 100$. The true rank is $R = 10$ and we scale the condition number of each factor matrix to 100.

For each tensor, we generate randomly 5 initialisations of factor matrices. Then we run each of the 4 algorithms $nb\_rand \times 5$ times.

The algorithms terminate either when the number of iterations exceeds $it\_max = 200$ or the minimal data fitting error f goes below $1.1 * \theta$. For (HER-)CPRAND, we compute an approximation error using $\hat{P} = 400$ samples, while the update steps use $|\mathcal{S}| = 10Rlog(R) \approx 231$ or $|\mathcal{S}_{small}| = 100$ samples.

We display boxplots of time, number of iterations, fit, score and percentage of restarts for HER-ALS and HER-CPRAND.

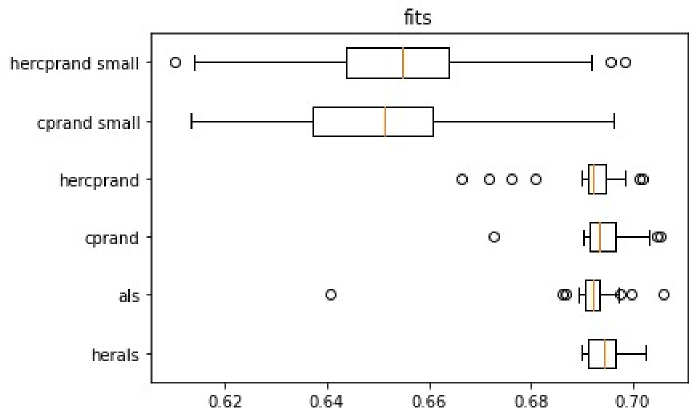


Fig. 5.10: fit

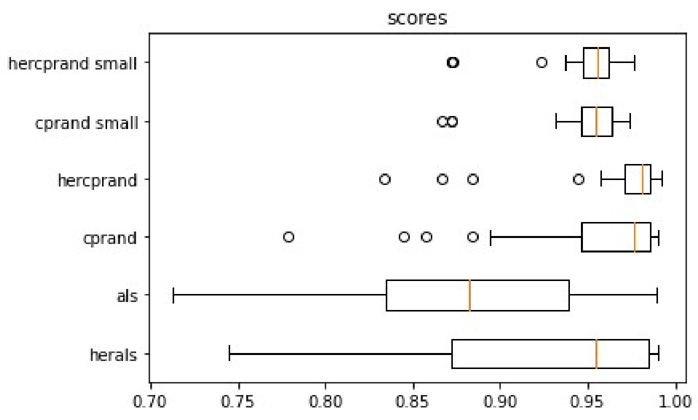


Fig. 5.11: score

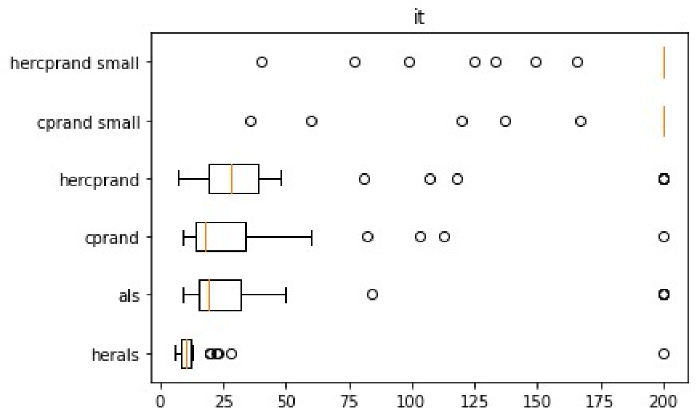


Fig. 5.12: it

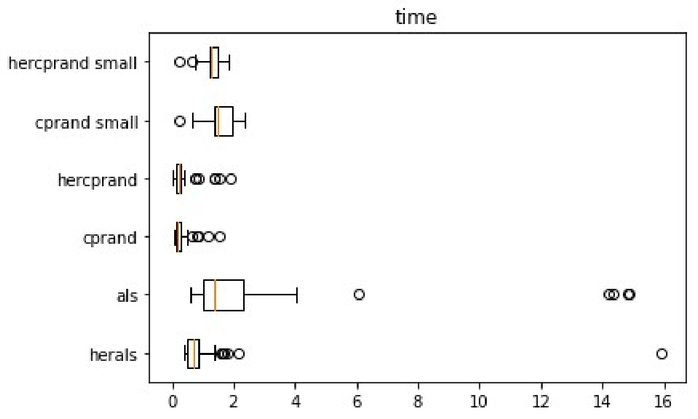


Fig. 5.13: time

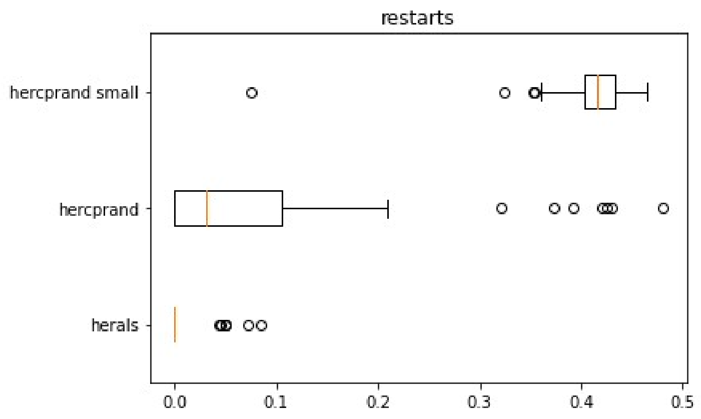


Fig. 5.14: restarts

From Fig.5.10, when $|\mathcal{S}|$ is small, the fit is worse, and we don't even have an improvement in time, because it requires more iterations since it makes less progress at each iteration.

In terms of fit, there is no significant difference between these 4 algorithms. But HER-CPRAND has the best score i.e. the biggest median value and the lowest variance. From Fig.5.12 HER-ALS needs less iterations than ALS because of the HER procedure. Even though, the randomized methods need more iterations in general than HER-ALS, the cost of each iteration is very low, so the randomized methods take less time. Comparing HER-CPRAND with CPRAND, even though we don't have an improvement in time, but the quality of factors' reconstruction is slightly better for HER-CPRAND.

For the restarts, HER-CPRAND do more restarts than HER-ALS due to the fact that we estimate the factors randomly. Note that for the bigger sample size, it is closer to HER-ALS.

We also test for the non-negative case. We take condition number = 10, $\theta = 0.1$, $tol = 1.05 * \theta$ and everything else unchanged.

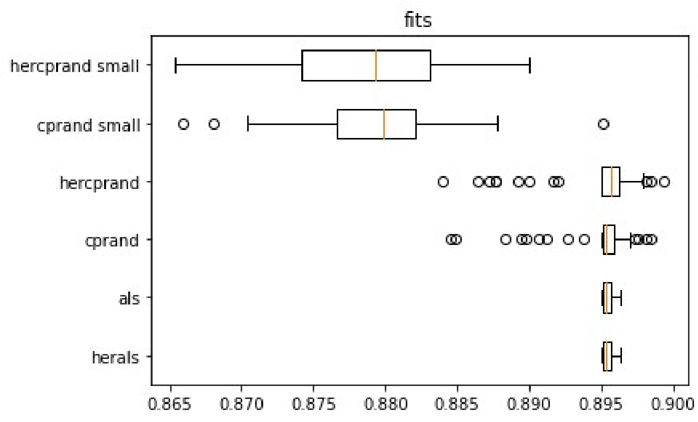


Fig. 5.15: fit

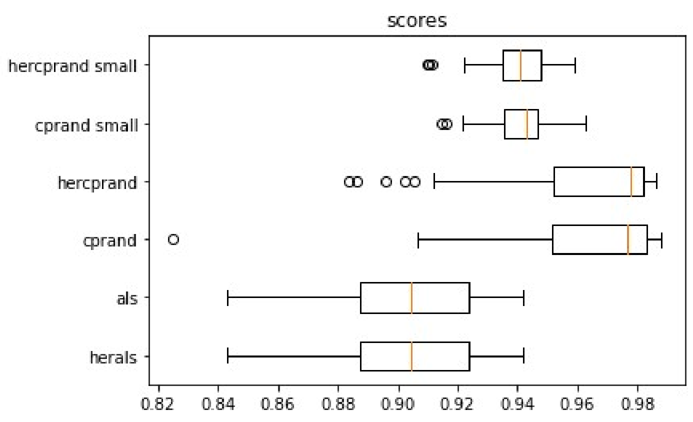


Fig. 5.16: score

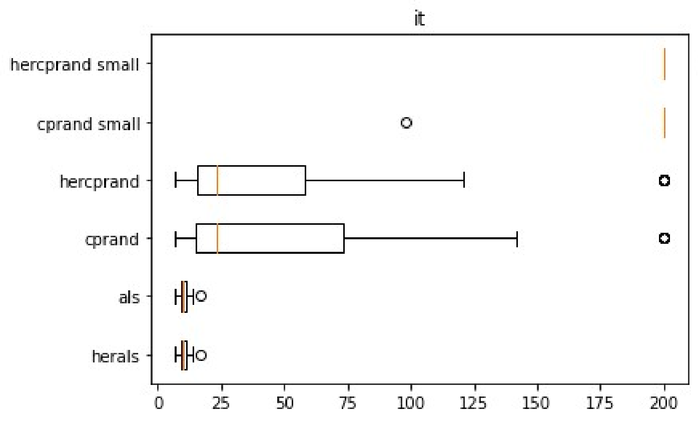


Fig. 5.17: it

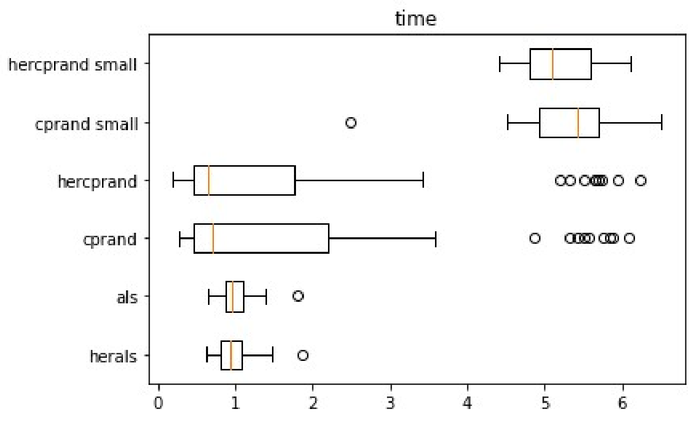


Fig. 5.18: time

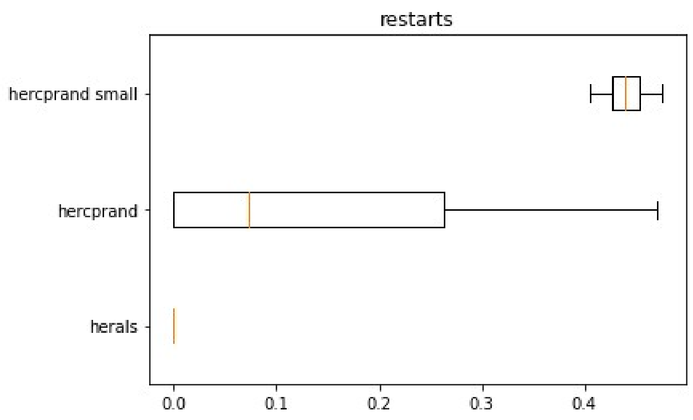


Fig. 5.19: restarts

(HER-)ALS work already well for the non-negative case. The score is better for (HER-)CPRAND, but there is no significant difference between CPRAND and HER-CPRAND. In Fig.5.18, we can see that HER-CPRAND is slightly better than CPRAND in terms of time, which is conflicting with Fig.5.9. Here we tolerate 5% error, while in the Fig.5.9 we tolerate 10%. One possible explanation is that HER-CPRAND allowing more accurate reconstructions, so it works better than CPRAND when the tolerance is smaller.

5.4 Real data

5.4.1 Fluorescence

Now we'd like to apply HER-CPRAND on a real data set in order to solve the source separation problem.

In fluorescence excitation emission spectroscopy, each sample is excited at J excitation wavelengths and the emission subsequently measured at K emission wavelengths. Hence, for I samples, an $I \times J \times K$ tensor $\mathcal{T}$ is obtained. If the samples contain, say R, chemical compounds that fluoresce, then the rank of the tensor should be R under ideal conditions up to the noise of the measurements.

The data set we used consists of five laboratory-made samples. Each sample contains different amounts of tyrosine, tryptophan and phenylalanine dissolved in phosphate buffered water. The samples were measured by fluorescence (excitation 250-300 nm, emission 250-450 nm, 1 nm intervals). The array to be decomposed is hence a non-negative 3-order tensor of size 5 × 51 × 201. Each substance gives a rank-one contribution to the data. In Fig.5.20 measurements of one of the samples are shown.

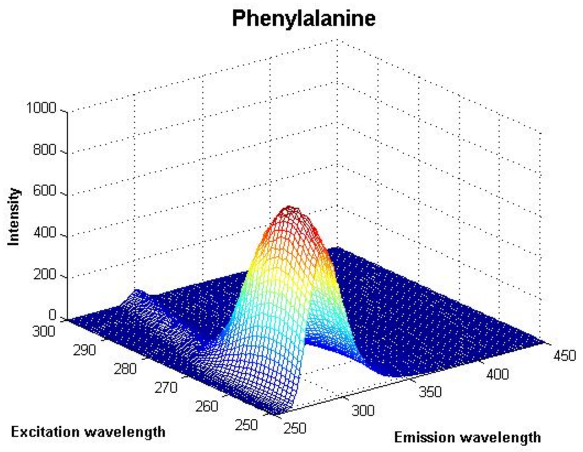


Fig. 5.20: Fluorescence landscape of a sample containing only phenylalanine - Rasmus Bro

Decomposition

We run NN-ALS and NN-HER-CPRAND with a random initialization of factors. We take $|\mathcal{S}| = 10Rlog(R)$ and $\hat{P} = 400$. The algorithms terminate if 100 iterations have elapsed, or $tol \leq 0.01$.

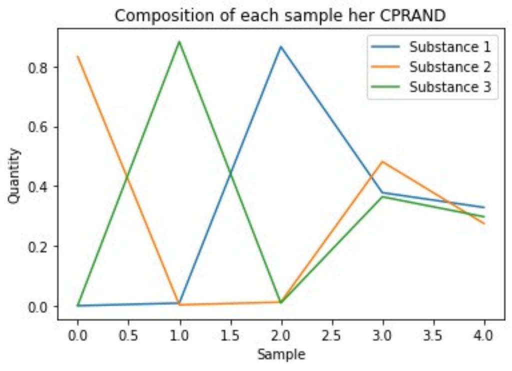


Fig. 5.21: Composition HER-CPRAND

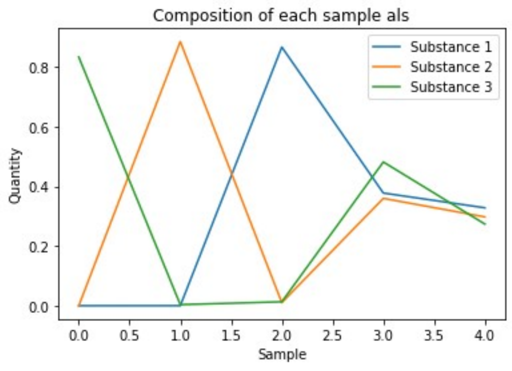


Fig. 5.22: Composition ALS

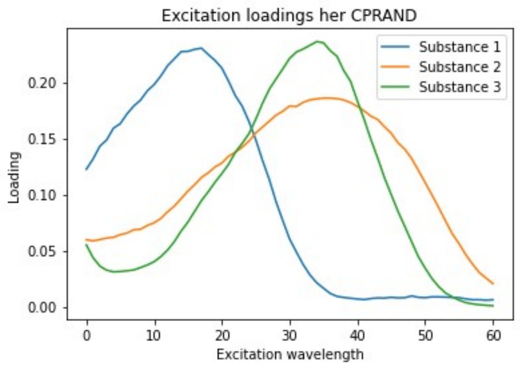


Fig. 5.23: Excitation HER-CPRAND

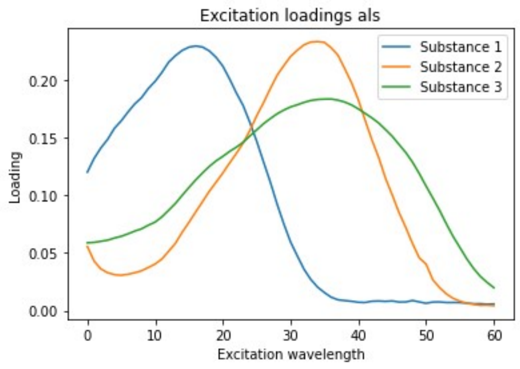


Fig. 5.24: Excitation ALS

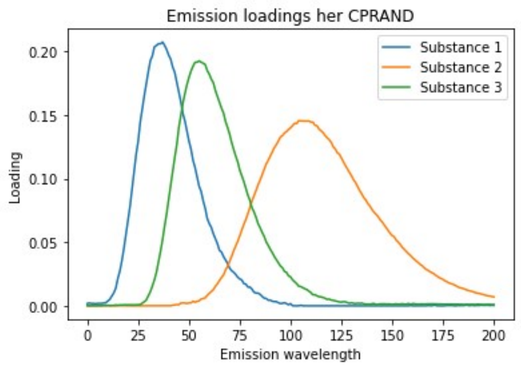


Fig. 5.25: Emission HER-CPRAND

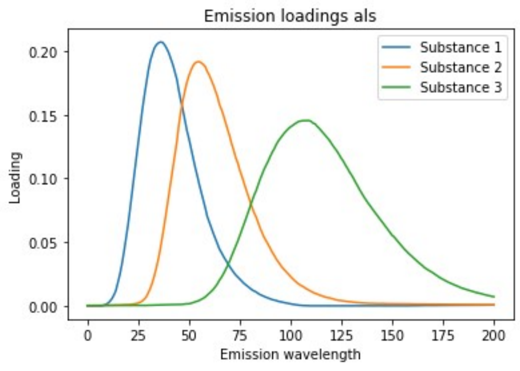


Fig. 5.26: Emission ALS

We obtain similar results with these 2 algorithms. The algorithms terminated because they have reached $it = 100$. With HER-CPRAND the factors are a little bit more noised since we do little improvement at each iteration.

Comparison

From the section 5.3, we didn't always have an improvement of HER-CPRAND compared to CPRAND for the non-negative case. We wonder whether it is the same for the real data set.

We run NN-(HER-)CPRAND with 5 different random initial factors, taking $|\mathcal{S}| = 10Rlog(R) \approx 33$, $|\mathcal{S}_{small}| = 10$ and $\hat{P} = 100$. The algorithms terminate if 200 iterations have elapsed, or $tol \leq 0.01$.

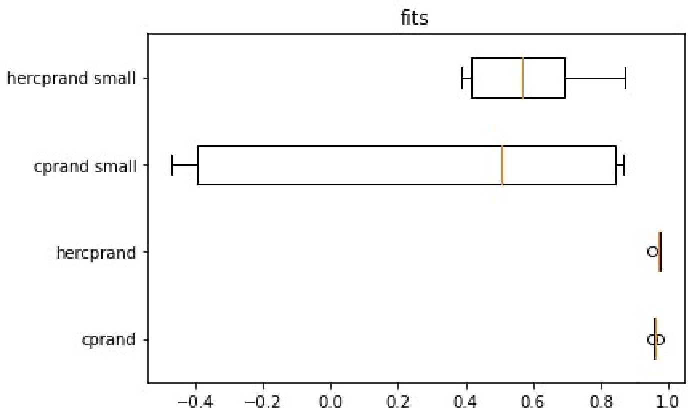


Fig. 5.27: fit

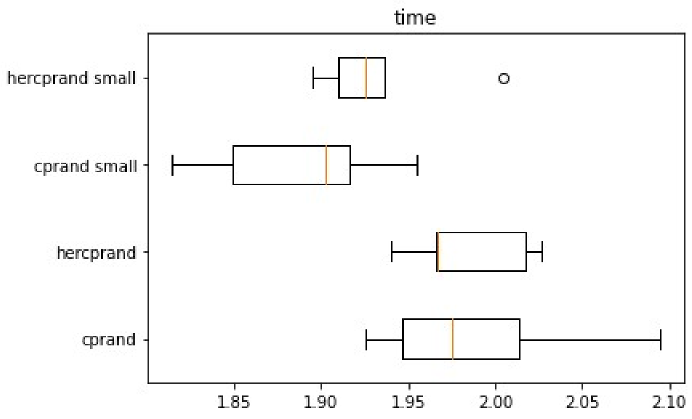


Fig. 5.28: time

The algorithms terminate because $it = 200$. We have a trade-off of speed and accuracy. In terms of fit and time, we can see that HER-CPRAND is slightly better than CPRAND.

5.4.2 Image

We test the performance of the algorithms on a hyperspectral image (HSI) Indian Pines.

This scene was gathered in North-western Indiana and consists of 145×145 pixels and 200 spectral reflectance bands. It forms a non-negative 3-order tensor with $R = 15$. The data is highly ill-conditioned. The condition numbers of the metricized pre-processed data tensor along all modes are [593; 642; 1009].[1]

We use 5 random initialization of factors, taking $|\mathcal{S}| = 10Rlog(R) \approx 407$, $|\mathcal{S}_{small}| = 200$ and $\hat{P} = 400$.

The algorithms terminate if 100 iterations have elapsed, or $tol \leq 0.08$.

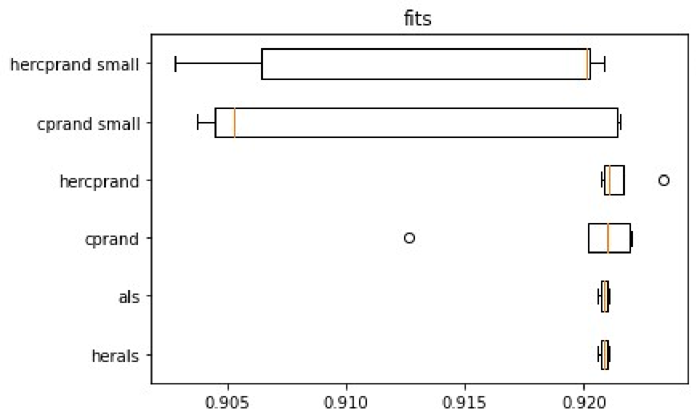


Fig. 5.29: fit

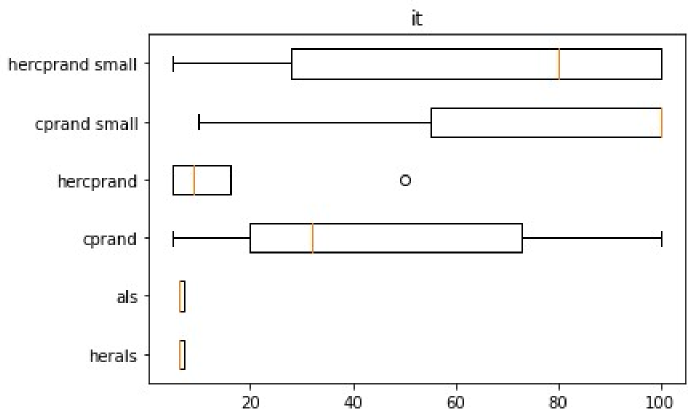


Fig. 5.30: it

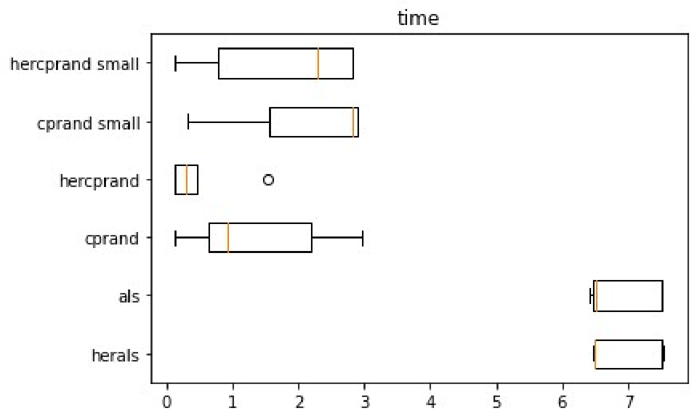


Fig. 5.31: time

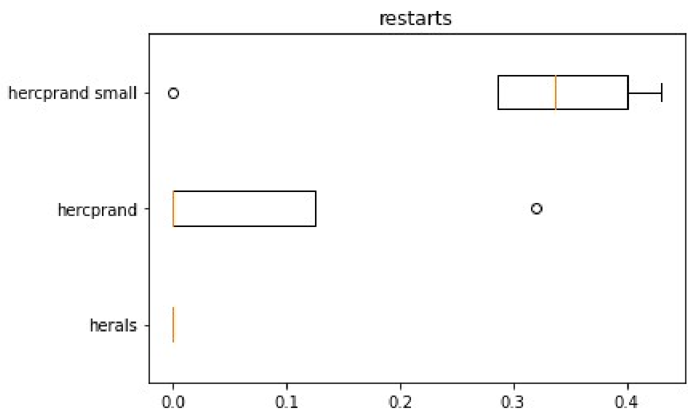


Fig. 5.32: restarts

In terms of fit, HER-CPRAND is a little bit better than CPRAND. With the HER procedure, the number of iterations decreases significantly (comparing HER-CPRAND with CPRAND). In terms of computation time, HER-CPRAND is the best algorithm.

6 Conclusion

We have seen that the new algorithm is faster than (HER-)ALS especially for big data, and it has the same or better decomposition quality.

Compared to CPRAND, even though the improvement in computation time is not significant, the quality of decomposition is in general slightly better. The performance of HER-CPRAND is higher in the highly ill-conditioned case e.g.image, with less computation time and better accuracy than CPRAND.

One possible improvement of the algorithm is to sample rows of the metricized tensor taking account of the ”importance” of the rows, the leverage scores, proposed by Larsen and Kolda in a recent work [6].

Another possible improvement is to use Andersen Acceleration for the extrapolation, i.e. instead of extrapolating with the last factors, we make a regression on the several latest factors to obtain an average.

With this project, I discovered the notion of low-rank tensor decomposition (especially the non-negative case), and other applications of tensor thanks to exchanges with people from Panama team. I also learned how to manage a long-time project, e.g. how to structure the code for a long-time project. I have acquired how to design tests thanks to the various tests developed. In the end, I had the experience of writing research documents, which requires more rigorousness, and at the same time, it is important to make things understandable (e.g. for a presentation).

Bibliography

[1] Le Khanh A. Man Shun Ang J. Cohen and N. Gillis. “Accelerating Block Coordinate Descent for Nonnegative Tensor Factorization”. In: *arXiv* (2021).

[2] Le Khanh A. Man Shun Ang J. Cohen and N. Gillis. “Extrapolated alternating algorithms for approximate Canonica Polyadic decomposition”. In: *HAL* (2019).

[3] G. Ballard C. Battaglino and T. G. Kolda. “A practical randomized CP tensor decomposition”. In: *SIAM Journal on Matrix Analysis and Applications* (2018), pp. 876–901.

[4] Le Khanh C. Jutten L. Duarte and S. Moussaoui. “Source separation in physical-chemical sensing”. In: *HAL* (2020).

[5] J. Cohen. “Nonnegative low rank approximations”. In: *iTWIST* (2020).

[6] Brett W Larsen and Tamara G Kolda. “Practical leverage-based sampling for low-rank tensor decomposition”. In: *arXiv* (2020).

[7] Brett W. Bader Tamara G. Kolda. “Tensor decompositions and applications”. In: *SIAM review* (2009), pp. 455–500.