RESEARCH

Optimization methods for tensor decomposition: a comparison of new algorithms for fitting the CP(CANDECOMP/PARAFAC) model

Huiwen Yu[1], Kasper Green Larsen[2] and Ove Christiansen[1*]

[*]Correspondence: ove@chem.au.dk
[1]Department of Chemistry, Aarhus University, Aarhus, Denmark
Full list of author information is available at the end of the article

Abstract

Tensor decomposition is widely used for multi-way data analysis and computations in chemical science. CP decomposition is one of the most useful tensor decomposition models for capturing the essential information in massive multi-way chemical data and for efficiently performing computations with such tensors. However, efficiently and accurately computing the tensor decomposition itself is a nontrivial problem that sometimes limits the advantage of tensor decomposition methods. In this work we propose and test three new decomposition algorithms, that are defined from extrapolation ideas applied to the alternating least square (ALS) algorithm for CP tensor decomposition. The performance of the proposed algorithms are validated on both a variety of simulated datasets and real experimental datasets including fluorescence spectroscopy data, hyperspectral data and electroencephalogram (EEG) data. The results show that the proposed algorithms significantly accelerate the standard CP-ALS decomposition while maintaining favorable accuracy. One of the proposed methods, denoted direct inversion of the iterative subspace-like extrapolated ALS(CP-AD), is inspired from widely used extrapolation procedures used in the context of solving non-linear equations in quantum chemistry, and shows a particular attractive combination of a much reduced number of iterations needed for convergence, and modest computational cost. For example, CP-AD provided resulting tensors of similar accuracy but significantly lower computational cost than the standard CP-ALS algorithm and the widely used line-search based CP-ALS extrapolation procedure. The proposed methodology may thereby boost the application of tensor decomposition modeling in both experimental and computational chemistry.

Keywords: Optimization; Tensor decomposition; Computations; Algorithms; Alternating least square

Introduction

With the developments in higher-order analytical instruments and the associated data collection and storage power of computers, the presence of large datasets is a common theme in chemical science. Furthermore, chemical data are not only measured or observed quantities from experimental chemistry, but may also be intermediate or final results of computer simulations. In many cases the data come in the format of a tensor, a multi-way array, which is a generalization of matrices to higher-order arrays. For example, fluorescence spectroscopy [1] and hyphenated chromatography-mass spectroscopy [2] involve the collection of datasets which come in the format of tensors. These massive tensor data are often not fully utilized and thoroughly investigated yet [3]. It remains a challenge for chemists to handle the large chemical tensor data and turn them into valuable chemical information [4, 5]. Developing efficient and accurate methods for manipulating and analyzing large tensor datasets is of pressing importance for obtaining insight and for the ability to carry out practical and accurate computations. Tensor decomposition is a promising and versatile tool for tensor data mining in many contexts, such as intelligent transportation system [6], biomedical analysis [7] and communications process analysis [8] etc. In recent years, tensor decomposition has gained more and more attention in chemistry. A number of recent works have shown the effectiveness of tensor decomposition techniques on multi-way chemical data analytics [9, 10, 11]. Tensors are also omnipresent in quantum chemistry. Many quantum chemical methods can be interpreted from a tensor decomposition perspective, i.e. the formally exact tensor is recast in terms of a format that can be considered a tensor-decomposition by ansatz. Furthermore, tensor decomposition can potentially boost methodology to make methods applicable that otherwise would not be due to tensors being too large or computations with them too costly[12].

CP(CANDECOMP/PARAFAC) decomposition is one of the most widely used tensor models [13]. It was first officially proposed by Harshman with the name of PARAFAC [14]. In the same year, the psychometricians Carroll and Chang [15] proposed the same tensor model but with a different name called CANDECOMP. CP refers to the combined name of CANDECOMP/PARAFAC. CP decomposition is very useful for representing the tensor in an attractive and economical format capturing the essential information with less data and more computationally convenient. For example, in the case of three-way tensor, CP decomposition factorizes the tensor into three factor matrices representing the information of each of the three ways. In other words, CP decomposition can be defined as the sum of outer products of three column vectors coming from these three factor matrices. However, the process of decomposing a given tensor adequately into simpler data objects is a non-trivial problem preventing taking full advantage of tensor decomposition methods in many contexts. The central challenge lies in efficiently and accurately computing the tensor decomposition itself.

Some algorithms have been developed for decomposing tensors into the CP format over the years, including the alternating least squares (ALS) algorithm [16], partitioned Hierarchical ALS algorithm [17], a randomized algorithm [18], damped Gauss Newton algorithm [19], and a proximal gradient algorithm [20], etc. Many of the available CP algorithms are computationally expensive and algorithmically complex, and some of them are not necessarily capable of computing an accurate decomposition in many natural settings. By far the most widely used workhorse algorithm for fitting a CP model is the ALS algorithm [21] (CP-ALS) which is normally employed as a standard algorithm in the available tensor toolboxes and as a benchmark algorithm for comparison. This is not only because the CP-ALS algorithm is simple and easy to implement in practice, but it also has flexibility for imposing meaningful constraints on the model and parallelizing the computation. The CP-ALS algorithm often represents a good trade-off in convergence time and solution quality compared to other algorithms [16]. However, the standard CP-ALS algorithm still often uses many iteration especially for large tensors. For complex data, it is not unusual that the CP-ALS algorithm cannot converge within an acceptable number of iterations and computational time. Local minima issues are another type of practical problems for the CP-ALS algorithm preventing the applicability of the CP model solution and the subsequent expert analysis in chemistry [22]. In many chemical applications, unique tensor solutions have vital chemical meanings. For example, when using a CP model on three-way fluorescence spectroscopy data, the tensor solutions, meaning the decomposed three factor matrices, represent for the pure emission spectra, excitation spectra, and concentrations of the compounds, thus the tensor solutions have to be uniquely determined. It is therefore highly timely and important to develop efficient and accurate alternatives to the CP-ALS algorithm.

In this work, we develop new algorithms that depart from CP-ALS by using extrapolation techniques to reduce the number of iterations and time consumption while maintain the accuracy of the solutions. The paper is structured in the following manner. In Section 2, we introduce the relevant methods and propose a set of accelerated CP-ALS algorithms. Section 3 briefly describes the experimental setups, simulated datasets and real datasets used for validating the algorithms. The experimental results are shown and discussed in Section 4. Finally, conclusions and perspectives are given in Section 5. Some additional results have been deferred to the additional files.

In this paper, we follow the conventional notations and nomenclature from [13, 23]. Scalars are denoted by italic non-bold letters e.g. S, while vectors are denoted by bold lowercase letters, e.g. $\mathbf{v}$, having elements v_i. In case of a matrix, bold capital letters such as $\mathbf{M}$ are used having elements M_{ij}. Tensors are denoted by bold capital letters with underline, e.g. $\underline{\mathbf{X}}$. The order of a tensor is the number of dimensions, and the individual dimensions are denoted as modes. We will focus on order three tensors. The element on the i_{th} row, j_{th} column and k_{th} slab in $\underline{\mathbf{X}}$ is denoted by x_{ijk}. The mathematical symbols $\odot$ represent the Khatri-rao product, while outer product and element-wise Hadamard product are denoted by $\circ$ and $\oplus$. The superscript T means the matrix transpose and superscript + means the Moore-Penrose pseudo inverse of a matrix. The norms used throughout are Frobenius norms (the square root of the sum of the absolute squares of the elements of vectors, matrices and tensors).

Methods

The CP-ALS algorithm

A given third-order tensor $\underline{\mathbf{X}} \in \mathbb{R}^{I\times J\times K}$, can be written in terms of a CP decomposition $\underline{\hat{\mathbf{X}}}$ and the corresponding residual $\underline{\mathbf{R}}$

$$\underline{\mathbf{X}} = \sum_{h=1}^{H} \mathbf{a}_h \circ \mathbf{b}_h \circ \mathbf{c}_h + \underline{\mathbf{R}} = \underline{\hat{\mathbf{X}}} + \underline{\mathbf{R}}. \tag{1}$$

Here $\mathbf{a}_h, \mathbf{b}_h$ and $\mathbf{c}_h$ are vectors, and H is the tensor rank (the number of rank-one tensors) of the CP representation. The residual is also often denoted by the error, and from it a scalar loss function can be obtained e.g. like the 2-norm of the residual tensor. The CP model can also be written in the stretched matrix form as follows:

$$\mathbf{X}_{I\times JK} = \mathbf{A}(\mathbf{C}\odot\mathbf{B})^T + \mathbf{R}_{I\times JK} \tag{2}$$

where $\mathbf{X}_{I\times JK}$ means the unfolded and stretched matrix from the third-order tensor $\underline{\mathbf{X}}$, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are factors matrices containing the $\mathbf{a}_h, \mathbf{b}_h$ and $\mathbf{c}_h$ as columns, and $\mathbf{R}_{I\times JK}$ refers to the residual in the same format as $\mathbf{X}_{I\times JK}$. The “outer product” of each term in Eq.(1) corresponding to outer products of columns from $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ is in general called a rank-1 tensor. The optimization problem of CP decomposition for a given target tensor $\underline{\mathbf{X}}$ and chosen rank can be written as minimizing the sum of the squared residual between the target tensor and the CP representation with respect to the factor matrices of the CP tensor

$$\min_{\mathbf{A},\mathbf{B},\mathbf{C}} ||\underline{\mathbf{X}} - \hat{\underline{\mathbf{X}}}|| = \min_{\mathbf{A},\mathbf{B},\mathbf{C}} ||\mathbf{R}|| \tag{3}$$

Thus the optimization problem corresponds to finding the factor matrices $(\mathbf{A}, \mathbf{B}, \mathbf{C})$ minimizing the residual. Unlike matrix decomposition, the CP decomposition is unique up to elementary scalings and permutations under mild conditions, see Ref. [13].

In CP-ALS, the estimated factor matrices are calculated iteratively in a least square way. In each iteration, the factor matrices are in turn determined in a least square manner while fixing the other two factor matrices in the case of third-order tensor. Thus, after updating the matrix $\mathbf{A}$ for the first mode, keeping $\mathbf{B}$ and $\mathbf{C}$ fixed, we proceed to the next modes, until all the three modes are traversed. The update process is then iterated until a predefined stop criterion, e.g. regarding the change in residual norm. In terms of equations the individual least square optimizations can be written as (see for example Ref. [13]): $\mathbf{A}_{estimate} = \mathbf{X}_{I\times JK}[(\mathbf{C}\odot\mathbf{B})^T]^+ = \mathbf{X}_{I\times JK}(\mathbf{C}\odot\mathbf{B})((\mathbf{C}\odot\mathbf{B})^T(\mathbf{C}\odot\mathbf{B}))^+$, where $(\mathbf{C}\odot\mathbf{B})^T(\mathbf{C}\odot\mathbf{B})$ can be calculated as $(\mathbf{B}^T\mathbf{B}) \oplus (\mathbf{C}^T\mathbf{C})$ [13, 24].

Bro’s Line Search Extrapolation

Bro’s line search extrapolation from the N-way toolbox, is one of the widely used extrapolation techniques to accelerate the convergence of the CP-ALS algorithm [25] and we therefore compare our methods to it. We shall denote it CP-AB in this work. Since CP-AB is not new with this work we only describe it very briefly and refer to the literature for more details. In order to get the best step size, CP-AB employs a golden section search procedure in the line search extrapolation, followed by a polynomial regression on the step size related parameters [25][26]. Specifically, it uses the golden section search to narrow down the range of acceleration step sizes on a predefined interval, and then a polynomial regression is fitted on the matrix containing all step size, their square numbers, constant number and residuals. By doing so, the best acceleration step size that produces the minimum residual at the current iteration can be obtained. The step size is then used for extrapolating the factors matrices $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$. The update process is repeated for each iteration until the CP-ALS algorithm converges. Basically, Bro’s line search extrapolation is deemed to be a robust method to accelerate the convergence of CP-ALS algorithm since the optimized step size is calculated and used at each iteration [26]. However, it is still computationally expensive, especially for large data sets.

Direct Inversion of the Iterative Subspace Extrapolated CP-ALS

Direct Inversion of the Iterative Subspace (DIIS) is a widely used and by now textbook method for solving various non-linear equations in quantum chemistry [27, 28], including for example the nonlinear coupled cluster amplitude equations [29]. The general idea in these applications of DIIS is that improved guesses of solutions are obtained by using information obtained during the previous few iterations. Here, we mimic the basic idea of the DIIS method and propose an as-simple-as-possible DIIS-like extrapolated CP-ALS algorithm which we shall denote CP-AD. To our knowledge, DIIS has not been explored at all before in a tensor decomposition context. In CP-AD, we define the residual tensor at iteration i as the difference between the target tensor $\underline{\mathbf{X}}$ and the current CP representation, $\underline{\mathbf{X}}_i$

$$\underline{\mathbf{R}}_i = \underline{\mathbf{X}} - \underline{\mathbf{X}}_i \tag{4}$$

We keep a set of factor matrices defining the CP decomposition at different iterations

$$\{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_n\}. \tag{5}$$

The set of all the factor matrices are represented in the vectors $\mathbf{f}_i$ consisting of stacked vectorized factor matrices (i.e. a vector consisting of the stacked vectorized factor matrices $vec(\mathbf{A})$, $vec(\mathbf{B})$, $vec(\mathbf{C})$ at iteration i, where vec is used to denote any consistent choice of vectorization of matrices or tensors). The residual are similarly stored as a set of vectorized residuals ($\mathbf{r}_i = vec(\mathbf{R}_i)$)

$$\{\mathbf{r}_1, \mathbf{r}_2, \cdots, \mathbf{r}_n\}. \tag{6}$$

Using this information from previous iterations we seek to obtain an improved guess from the last iteration. This can be considered as some kind of extrapolation or as optimization. To proceed, we assume a linear relationship between the factor matrix elements defining the approximate tensor in a given iteration and the residual. In fact, the estimated decomposed tensor, and therefore also the residual, depends on the product of factor matrix elements according to the definition in Eq.(1) and the relation is therefore non-linear. However, given that the factor matrices are not too far from optimal, the change in residual from the optimal values for the given rank, may to a reasonable approximation be considered linear to the deviations of the factor matrices from their optimal values.

Based on this assumed linearity, CP-AD tries to estimate the best linear combination of the available factor matrix elements. Thus, the optimal parameters are written as a correction to the parameters of the last iteration with weights for the difference between this and previous parameter sets

$$\mathbf{f}_{n,opt} = \mathbf{f}_n + \sum_{i=1}^{n-1} c_i(\mathbf{f}_i - \mathbf{f}_n) = \sum_{i=1}^{n} c_i \mathbf{f}_i \tag{7}$$

The last equality follows with the following constraint on the weights c_i, $\sum_{i=1}^{n} c_i = 1$. The weights c_i are to be determined as the set giving the expected lowest residual norm of the corresponding optimized tensor while satisfying this constraint. Referring to the assumed linearity, the residual of the optimal parameters is estimated as $\sum_{i=1}^{n} c_i \mathbf{r}_i$. To perform the constrained optimization of the norm square of this, we use a Lagrange method. Specifically, we construct the following Lagrangian function (see also [27])

$$L = \sum_{i,j=1}^{n} c_i \mathbf{M}_{ij} c_j - 2\lambda(\sum_{i=1}^{n} c_i - 1) \tag{8}$$

where $\mathbf{M_{ij}} = \mathbf{r}_i^T \mathbf{r}_j$, λ is an undetermined Lagrangian multiplier and the factor of 2 is introduced for convenience. The minimization of the Lagrangian L with respect to the coefficients and multipliers gives the following linear equations

$$\begin{pmatrix} \mathbf{M} & -\mathbf{1} \\ -\mathbf{1} & 0 \end{pmatrix} \begin{pmatrix} \mathbf{c} \\ \lambda \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ -1 \end{pmatrix} \tag{9}$$

determining the weights c_i. From these, we obtain $\mathbf{f}_{n,opt}$ through Eq.(7), which defines updated factor matrices. We can then use it to compute the actual residual, $\underline{\mathbf{R}}_{n,opt}$, for the extrapolated tensor. We have integrated the above optimization with ALS, such that after an ALS round of factor matrix updates for all modes, we perform the above DIIS-inspired analysis for the recent factor matrices and tensors to produce an optimized DIIS solution. We have implemented two algorithmic variations of this. In the first algorithm, if $\mathbf{f}_{n,opt}$ has smaller residual norm than the previous, we will keep $\mathbf{f}_{n+1} = \mathbf{f}_{n,opt}$ and $\underline{\mathbf{R}}_{n+1} = \underline{\mathbf{R}}_{n,opt}$ and proceed with it for the next iteration. However, if it is found at some stage that the optimization step leads to an increasing residual, we abort the optimization and give up the extrapolated estimates and residuals. In that case, we will continue with the standard least square updates of that iteration. In the second implementation of DIIS based CP, we always continue with the DIIS extrapolated estimates for the next iteration without any checking procedure. Therefore, we name the first implementation as conservative CP-AD (CP-ADc) and the second implementation as simply CP-AD. In the main text of the paper, we show the algorithm (Algorithm 1) and results of CP-AD since it was found to be more efficient and sufficiently stable in many of our exploratory computations. The algorithm details of CP-ADc is shown in the additional files in the supporting information. For computational efficiency it is attractive to limit the subspace in CP-AD to only the latest few estimates of $\mathbf{f}$ in the iterative procedure, n_{trial}. I.e. in the sums in Eqs.(7-8) we need not sum over all iterations, but limit ourselves to the most recent iterations that is anticipated to be most relevant anyway. In our case, we use the last three iterations ($n_{trial} = 3$), and it yields favourable results. Similarly, we start with n_{trial} standard ALS iterations before applying the DIIS-type optimizations.

Algorithm 1 CP-AD algorithm

```
1: initialization: Do SVD on raw tensor $\underline{\mathbf{X}}$ to get a set of initial factors matrices (A,B,C), and set $i = 1$
2: while $i = 1, 2, 3, ...$ do
3:     update the least square factor matrices - first use CP-ALS:
4:     $\mathbf{A} = \mathbf{X}_{I\times JK}[(\mathbf{C}\odot\mathbf{B})((\mathbf{B}^T\mathbf{B})\oplus(\mathbf{C}^T\mathbf{C}))^{+}]$;
5:     $\mathbf{B} = \mathbf{X}_{J\times IK}[(\mathbf{C}\odot\mathbf{A})((\mathbf{A}^T\mathbf{A})\oplus(\mathbf{C}^T\mathbf{C}))^{+}]$;
6:     $\mathbf{C} = \mathbf{X}_{K\times IJ}[(\mathbf{B}\odot\mathbf{A})((\mathbf{A}^T\mathbf{A})\oplus(\mathbf{B}^T\mathbf{B}))^{+}]$;
7:     calculate the residual tensor $\underline{\mathbf{R}}_i = \underline{\mathbf{X}} - \underline{\mathbf{X}}_i$
8:     if $i \le n_{trial}$ then
9:         store the factor matrices and the stacked residual vector
10:    else
11:        Perform CP-AD acceleration:
12:        store the factor matrices and the stacked residual vector
13:        Use the last $n_{trial}$ residuals to compute the $\mathbf{M}$ matrix,$\mathbf{M}_{ij} = \mathbf{r}_i^T\mathbf{r}_j$
14:        Calculate the weights of the last $n_{trial}$ factor matrix estimates according to equation 9
15:        compute improved factor matrices according to equation 7 to get optimized $(\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i)$
16:        calculate the residual tensor $\underline{\mathbf{R}}_i = \underline{\mathbf{X}} - \underline{\mathbf{X}}_i$
17:    end if
18:    set $i = i + 1$
19:    repeat until the residual change reaches the predefined criterion
20: end while
```

Geometric Search Extrapolated CP-ALS

Geometric Search was initially proposed for computing the PARAFAC2 decomposition, which is a non-strict-multilinear model for analyzing e.g. shifted multi-way GC-MS data [26]. In this paper, we extend the Geometric Search scheme and combine it with PLS toolbox line search [25] (Eigenvector Research, Inc., Manson, WA USA) to extrapolate the CP-ALS algorithm. We name the new algorithm CP-AG. In a number of successive ALS iterations, the estimated parameters D_i in the i_{th} ALS iteration are supposed to be expressed as a sequence:

$$D_i = ar^i + b, i_1 \le i \le i_2, 0 \le r \le 1 \tag{10}$$

where i_1 and i_2 are the starting point (iteration) and ending point (iteration) of the considered sequence, and r is the so-called convergence rate. The parameters $\{D_i\}$ include the factor matrix parameters in $\mathbf{A}$, $\mathbf{B}$, and, $\mathbf{C}$ obtained from the full ALS iterations. a and b are parameters, where it is particularly noteworthy that increasing the sequence it converges to b, which forms the basis for an extrapolation algorithm [26]. Rather than using a full ALS sequence one can use a few parameters obtained during a few ALS iterations (here four). We accordingly follow steps 3-6 of Ref.[26] in implementing CP-AG and the full algorithm is documented in Algorithm 2. As it turns out it is less attractive than other extrapolation algorithms in many cases we avoid repeating any details and refer to the previous work and the algorithm for all details.

Algorithm 2 CP-AG algorithm

```
1: initialization: Do SVD on raw tensor $\underline{\mathbf{X}}$ to get a set of initial factors matrices (A,B,C), and set
   $i = 1$, confidence threshold $= 0.9$, probability percentage $P = 0.7$, $flag = 0$
2: while $i = 1, 2, 3, ...$ do
3:     if $flag = 0$ then
4:         update the least square factor matrices:
5:         $\mathbf{A} = \mathbf{X}_{I\times JK}[(\mathbf{C}\odot\mathbf{B})^T]^+$;
6:         $\mathbf{B} = \mathbf{X}_{J\times IK}[(\mathbf{C}\odot\mathbf{A})^T]^+$;
7:         $\mathbf{C} = \mathbf{X}_{K\times IJ}[(\mathbf{B}\odot\mathbf{A})^T]^+$;
8:         where
9:         $[(\mathbf{C}\odot\mathbf{B})^T]^+ = (\mathbf{C}\odot\mathbf{B})((\mathbf{B}^T\mathbf{B})\oplus(\mathbf{C}^T\mathbf{C}))^+$;
10:        $[(\mathbf{C}\odot\mathbf{A})^T]^+ = (\mathbf{C}\odot\mathbf{A})((\mathbf{A}^T\mathbf{A})\oplus(\mathbf{C}^T\mathbf{C}))^+$;
11:        $[(\mathbf{B}\odot\mathbf{A})^T]^+ = (\mathbf{B}\odot\mathbf{A})((\mathbf{A}^T\mathbf{A})\oplus(\mathbf{B}^T\mathbf{B}))^+$;
12:        calculate the residual tensor $\underline{\mathbf{R}}_i = \underline{\mathbf{X}} - \underline{\mathbf{X}}_i$
13:    else if $flag = 1$ then
14:        compute $r = (D_i - D_{i-1})/(D_{i-1} - D_{i-2})$ where $D$ refers to the factor parameters in
   $(\mathbf{A}, \mathbf{B}, \mathbf{C})$
15:        if $0 \le r \le 1$ then
16:            do geometric search extrapolation by $D_t = (D_{i-2}D_i - D_{i-1}^2)/(D_{i-2} + D_i - 2D_{i-1})$
17:        else if $r < 0$ or $r > 1$ then
18:            do PLS toolbox line search by $D_t = D_i + i^{1/2} \times (D_i - D_{i-1})$
19:        end if
20:        calculate the residual tensor norm $R_t = ||\underline{\mathbf{R}}_t||$ and evaluate the extrapolation:
21:        if $R_t \le R_{i-1}$ then
22:            $(\mathbf{A_i}, \mathbf{B_i}, \mathbf{C_i}) \leftarrow D_t$
23:            flag=0
24:        else
25:            $\mathbf{A_i} \leftarrow \mathbf{A_{i-1}}$  $\mathbf{B_i} \leftarrow \mathbf{B_{i-1}}$  $\mathbf{C_i} \leftarrow \mathbf{C_{i-1}}$
26:            flag=2
27:        end if
28:    else if flag=2 then
29:        do PLS toolbox line search by $D_t = D_i + i^{1/2} \times (D_i - D_{i-1})$
30:        calculate the residual tensor norm $R_t = ||\underline{\mathbf{R}}_t||$ and evaluate the extrapolation:
31:        if $R_t \le R_{i-1}$ then
32:            $(\mathbf{A_i}, \mathbf{B_i}, \mathbf{C_i}) \leftarrow D_t$
33:        else
34:            $\mathbf{A_i} \leftarrow \mathbf{A_{i-1}}$  $\mathbf{B_i} \leftarrow \mathbf{B_{i-1}}$  $\mathbf{C_i} \leftarrow \mathbf{C_{i-1}}$
35:            increase the denominator of the power of $i$ with 1
36:        end if
37:        flag=0
38:    end if
39:    store the maximum parameters from each column of factors matrices for each iteration
40:    take the parameters of four iterations and conduct the line regression on each parameter
41:    if coefficient of determination $R^2$ of $P*$ total number of parameters $\ge confidence$ then
42:        flag=1
43:    else
44:        flag=0
45:    end if
46:    set $i = i + 1$
47:    repeat until the residual change reaches the predefined criterion
48: end while
```

Optimized Nesterov-like Extrapolated CP-ALS

Nesterov extrapolation is a well-established acceleration method for gradient descent. It was firstly proposed by Nesterov for minimizing convex problems and was confirmed to achieve the best convergence performance [30]. By introducing and correcting the momentum term, Nesterov's acceleration scheme provides a larger movement and a flexible correction to the descent gradient so that gains in convergence is achieved. Recently, Ang et al. mimicked the Nesterov acceleration scheme and proposed a Nesterov-like extrapolation scheme for accelerating the convergence of non-negative matrix factorization [31] and further extended it to accelerate non-negative Hierarchical ALS algorithm (CP-HALS) for tensor decomposition [32]. Here we will focus on CP-ALS with the relative disadvantages of CP-HALS discussed by [24]. Thus, we develop an Optimized Nesterov-like extrapolation scheme for accelerating the convergence of the CP-ALS algorithm, named as CP-AO. In CP-AO, we implement a new correction scheme on the extrapolation variable β in the context of CP-ALS algorithm. In greater detail, CP-ALS iteratively updates the matrices $\mathbf{A}, \mathbf{B}$ and $\mathbf{C}$ by solving a least squares problem. If CP-ALS would update $\mathbf{A}_{\text{old}}$ to $\mathbf{A}_{\text{new}}$, we instead update to $\mathbf{A}_{\text{old}} + \beta(\mathbf{A}_{\text{new}} - \mathbf{A}_{\text{old}})$ for some step size/the extrapolation parameter β that depends on the residual errors and other control parameters.

CP-AO starts with an initialization of control parameters $\bar{\beta}\in[0,1]$, $\beta\in[0,1]$, $1\leq\bar{\eta}\leq\eta\leq\in\gamma$. When the residual increases, β from the current iteration will be decreased by parameter γ and it will be used as the ceiling parameter $\bar{\beta}$ for the next iteration. As long as the the residual decreases, the parameter β will be increased by taking the maximum value from $\bar{\beta}$ and $\eta\beta$ and the ceiling parameter will also be increased by taking the maximum value from one and $\bar{\eta}\bar{\beta}$, which means that we are confident to permit the algorithm to move a large step if the current extrapolation is accepted. In our tests, the combination of γ=1.5, $\bar{\eta}$=1.03, η=1.1, β=0.05 and $\bar{\beta}$=1 yields favorable results. More details of CP-AO can be obtained from Algorithm 3.

Algorithm 3 CP-AO algorithm

```
1: initialization: Do SVD on raw tensor $\underline{\mathbf{X}}$ to get a set of initial factors matrices (A,B,C), and set
   $i = 1$, $\gamma$=1.5, $\bar{\eta}$=1.03, $\eta$=1.1, $\beta$=0.05 and $\bar{\beta}$=1
2: while $i = 1, 2, 3, \ldots$ do
3:     update the factor matrices:
4:     $\mathbf{A} = \mathbf{X}_{I\times JK}[(\mathbf{C}\odot\mathbf{B})^T]^+$;
5:     $\mathbf{B} = \mathbf{X}_{J\times IK}[(\mathbf{C}\odot\mathbf{A})^T]^+$;
6:     $\mathbf{C} = \mathbf{X}_{K\times IJ}[(\mathbf{B}\odot\mathbf{A})^T]^+$;
7:     where
8:     $[(\mathbf{C}\odot\mathbf{B})^T]^+ = (\mathbf{C}\odot\mathbf{B})((\mathbf{B}^T\mathbf{B})\oplus(\mathbf{C}^T\mathbf{C}))^+$;
9:     $[(\mathbf{C}\odot\mathbf{A})^T]^+ = (\mathbf{C}\odot\mathbf{A})((\mathbf{A}^T\mathbf{A})\oplus(\mathbf{C}^T\mathbf{C}))^+$;
10:    $[(\mathbf{B}\odot\mathbf{A})^T]^+ = (\mathbf{B}\odot\mathbf{A})((\mathbf{A}^T\mathbf{A})\oplus(\mathbf{B}^T\mathbf{B}))^+$;
11:    do CP-AO extrapolation to get $(\mathbf{A_t}, \mathbf{B_t}, \mathbf{C_t})$:
12:    $\mathbf{A_t} = \mathbf{A} + \beta * (\mathbf{A} - \mathbf{A_{old}})$
13:    $\mathbf{B_t} = \mathbf{B} + \beta * (\mathbf{B} - \mathbf{B_{old}})$
14:    $\mathbf{C_t} = \mathbf{C} + \beta * (\mathbf{C} - \mathbf{C_{old}})$
15:    calculate the residual tensor norm $R_i = ||\underline{\mathbf{R}}_i|| = ||\underline{\mathbf{X}} - \underline{\mathbf{X}}_i||$ and evaluate the extrapolation:
16:    if $R_i \leq R_{i-1}$ then
17:        $\mathbf{A_{old}} \leftarrow \mathbf{A}$ $\mathbf{B_{old}} \leftarrow \mathbf{B}$ $\mathbf{C_{old}} \leftarrow \mathbf{C}$
18:        $\mathbf{A} \leftarrow \mathbf{A_t}$ $\mathbf{B} \leftarrow \mathbf{B_t}$ $\mathbf{C} \leftarrow \mathbf{C_t}$
19:        set $\beta = max(\bar{\beta}, \beta * \eta)$
20:        set $\bar{\beta} = max(1, \bar{\beta} * \bar{\eta})$
21:    else
22:        $\mathbf{A_{old}} \leftarrow \mathbf{A}$ $\mathbf{B_{old}} \leftarrow \mathbf{B}$ $\mathbf{C_{old}} \leftarrow \mathbf{C}$
23:        $\mathbf{A} \leftarrow \mathbf{A}$ $\mathbf{B} \leftarrow \mathbf{B}$ $\mathbf{C} \leftarrow \mathbf{C}$
24:        set $\bar{\beta} = \beta$
25:        set $\beta = \beta/\gamma$
26:    end if
27:    set $i = i + 1$
28:    repeat until the residual norm change reaches the predefined criterion
29: end while
```

Experiments setups and data sets

Experimental setups

All simulations and calculations were performed using an 1.8 GHz Dual-Core Intel Core i5 computer running macOS version 12.6.2 and MATLAB R2018a (The MathWorks, Inc., Natick, MA USA). The maximum number of iterations for all the algorithms was set to a very large number (5000) but was never met in any of our computations. Singular value decomposition solution of the unfolded tensor is used for the initialization for all the algorithms. In order to make a fair comparison, we implemented all algorithms from scratch to avoid relying on library implementations that e.g. support unnecessary functionality.

Simulated data set

A set of tensors with different sizes, different levels of noise and different number of ranks are simulated. The factor matrices **A**, **B** and **C** used for simulating the tensor contains uniformly distributed random numbers in the interval [0,1], respectively. For balanced size tensors, the size of all the factor matrices are set as $150 \times H$. For imbalanced size tensors, the sizes of **A**, **B** and **C** are set to $250 \times H$, $1000 \times H$ and $10 \times H$. After a set of operations on **A**, **B** and **C** according to the CP decomposition equations, the tensors with the balanced size $150 \times 150 \times 150$ and unbalanced size $250 \times 1000 \times 10$ are finally simulated. In order to make the simulation more realistic, we add 0.1%, 1% and 10% levels of noise to each entry of the simulated tensors, respectively. The rank is simulated as 5, 6 and 7 respectively, for both balanced size tensors and imbalanced size tensors. For each combination of noise level and rank complexity, 10 tensors are simulated with balanced size and imbalnced size. A total of 180 tensors (2*3*3*10) are simulated, leading to almost 1000 CP/PARAFAC models that are used for the further analysis. Local minima are very rare but checked for in all cases (by comparison of match with original tensor and decompositions obtained from different algorithms) and if found the local minima solutions are removed from the statistical analysis.

Real data set: Fluorescence spectroscopy data

The fluorescence spectroscopy data set measures the metabolite of human blood plasma samples from a control study in colorectal cancer. Excitation Emission Matrix (EEM) of 289 samples are recorded with the emission from 300 to 600 nm by every 1 nm and with the excitation from 250 to 450 nm by every 5 nm, which results in a $289 \times 301 \times 41$ third-order tensor representing the dimensions of samples, emission spectra and excitation spectra. The NaN elements in the fluorescence tensor are replaced by zero values and three CP components are used to fit the tensor in the models. More details about the fluorescence data set can be obtained from the original reference [33].

Real data set: Hyperspectral data

The hyperspectral data is a portion of Salinas A-scene data set which is a subscene of the huge Salinas images. The data is acquired by using the airborne visible/infrared imaging spectrometer over Salinas Valley. It consists of 80×84 pixels located within the same scene and includes six end−members. Thus, a rank-six CP model can be fitted to the hyperspectral tensor structured with the size of $80 \times 84 \times 204$. Regarding more information about the hyperspectral data set and the full images, see [34][35].

Real data set: EEG data

The event-related electroencephalogram (EEG) data exam the event-related potentials and are used as the indices of stimulus elicited activity. This data set contains the inter-trial phase coherence (ITPC) of 14 individuals. The data are collected during a proprioceptive pull of left and right hand of 14 individuals, so a total of 28 measurements are recorded. It is measured as multi-channel signals with 64 scalp electrodes/channels. The signals expressed in time and frequency domains are vectorized into vector of length 4392. The dataset is wavelet transformed and contains only the pure ITPC activity signals. Thus, it leads to a third-order tensor with the size of $4392 \times 64 \times 28$ representing the dimension of time-frequency domain, channels and measurements. The complex numbers in the tensor are transformed into real numbers by calculating the absolute values of the elements. A rank−three CP model is fitted to the data [36]. The data set is available from the ERPWAVELAB [37][38].

Results and discussions

Simulated data sets

The results of the residual evolution on rank five-simulated datasets with 10% noise level and balanced size and imbalanced size are shown on Figure 1 and Figure 2 for the first examples of simulated tensors. Overall, the proposed extrapolated algorithms successfully speed up the convergence of CP decomposition on both two types of simulated datasets. In terms of time consumption, CP-AO, CP-AG and

CP-AD manage to decrease the time of the CP-ALS algorithm and are faster than CP-AB on two simulated datasets. For the data with balanced size, CP-AO is the fastest and it is almost three times faster than CP-AB, followed by CP-AD and CP-AG. For the data with imbalanced size, CP-AD and CP-AO are the fastest and they are more than four times faster than CP-AB, followed by CP-AG. The standard CP-AB takes more time than CP-ALS in the case of simulated data with imbalanced size, meaning it happens that CP-AB sometimes fails to accelerate the convergence of CP-ALS algorithm. In terms of convergence robustness, the CP-AD and CP-AB share a common robust convergence behavior, indicating by fairly monotonic evolution residual norm curves. Conversely, the convergence process of CP-AO and CP-AG present the characteristic of oscillatory descent by a set of jumping behaviors. Regarding the number of iteration in the algorithms, all the extrapolated algorithms successfully decrease the number of iterations for convergence. Particularly, CP-AD achieve the best performance and the number of iteration it takes to converge is almost six times less than the ones of CP-ALS in the case of simulated data with imbalanced size. For the simulated data with balanced size, CP-AB and CP-AD are almost the same in terms of number of iterations. The results on the other simulated datasets with 10% noise level are in similar pattern with the ones from presented datasets and the results are shown in the supplementary information of the paper (see figures S1-S4.)

The details of final results of all the algorithms on all the simulated datasets are shown in Table 1 and Table 2. Since the real factor matrices of the simulated datasets are known, we measure the similarity between factor matrices from the CP algorithms and the corresponding real factor matrices by calculating the Tucker congruence coefficient[39]. The so-called Tucker congruence coefficient measures the cosine of the angle between any pair of vectors and it varies from -1 to 1. In our calculations, we take the average Tucker congruence coefficient value over pairs of columns vectors for each comparison. The used Tucker congruence coefficients in the table are the average number of all the ten tensors model results. Time consumptions and number of iterations of the algorithms are calculated in the same manner, being the average number of model results from ten tensors.

The highest Tucker congruence coefficient is one meaning the two matrices are exactly the same. On the contrary, a Tucker congruence coefficient of zero is not favorable since means the two matrices are dissimilar. Due to the sign ambiguity and column ambiguity problems in CP decomposition, one has to be careful when calculating the Tucker congruence coefficient. It is observed from Table 1 that the Tucker congruence coefficients of factor matrices from proposed extrapolated algorithms are all one(round to two decimal places), meaning that the decomposed factor matrices are virtually the same as the real factor matrices used for simulating the balanced size tensors. Similarly, the decomposed factor matrices from the proposed algorithms are also the same as the real factor matrices used for simulating the imbalanced size tensors, which is indicated by Tucker congruence coefficients as one. This is consistent for the tensors with different extents of rank complexity and different levels of noise. Thus, it means that accurate tensor decomposition solutions are reached by the proposed tensor decomposition algorithms. Moreover, it can be again seen that CP-AB is advantageous in decreasing the number of iterations on the simulated datasets even though the time consumption is high. Thus the time cost per iteration of CP-AB is high compared to the proposed extrapolated CP-ALS algorithms. It is evident from the table that the proposed CP-AD outperforms all the other algorithms in terms of consumed number of iterations and time consumption, taking the fewest number of iterations to converge on the simulated datasets and, together with CP-AO, giving the fastest computational time. For instance, CP-AD averagely takes more than six times fewer iterations to converge than CP-ALS on balanced size tensors with rank six and 0.1% noise level. The advantageous patterns holds on for all the tensors with different levels of noise and different extents of rank complexity. Compared to CP-AB, CP-AG and CP-AO are also advantageous in terms of average time consumption and, for CP-AO, consumed number of iterations. However, they either averagely takes higher number of iterations to converge (CP-AO) or takes both higher number of iterations and longer time to converge (CP-AG) comparing to CP-AD. Recalling that of course compute timings may vary somewhat with details of implementation and the computing environment, and other factors, but are still an overall loose indicator of efficiency. The differences in number of iterations is a more strict factor in evaluating relative merits of algorithms. In this case they together gives CP-AD as the best performing among all the studied CP algorithms for these random tensors. We also explore performance of the algorithms in the case of overfitting in CP modelling. For the balanced size tensors with 10% noise level, we calculate the CP models with real rank number plus one (overfitting), e.g. using rank six model to compute rank five tensor. The results comes out that CP-AD takes 31 iterations to converge on average being the fastest algorithm, while CP-AO takes 81 iterations, CP-AG takes 55 iterations, CP-AB takes 66 iterations and CP-ALS takes 107 iterations to converge. Again, CP-AD takes the smallest number of iterations to converge among all the CP algorithms.

Real datasets: Fluorescence spectroscopy data

The results of proposed algorithms on three-way fluorescence spectroscopy data are displayed on Figure 3. Similar to the results of simulated datasets, the proposed

extrapolated algorithms CP-AD and CP-AG succeed in accelerating the convergence of the CP-ALS algorithm and they are all faster than the standard CP-AB algorithm on this fluorescence spectroscopy data. The CP-AO gets stuck in a local minimum solution on this data indicated by a lower explained fit percentage compared to the global minimum solution from the other algorithms. The explained fit percentage means the percentage variation explained by the CP model defined as $1 - (||\underline{\mathbf{X}}||^2 - ||\hat{\underline{\mathbf{X}}}||^2)/||\underline{\mathbf{X}}||^2$ where $\underline{\mathbf{X}}$ is the tensor and $\hat{\underline{\mathbf{X}}}$ is the approximation. Thus the results of CP-AO on this fluorescence dataset is not taken into account for the analysis. In terms of timing, the proposed CP-AD and CP-AG almost behave equally well and their timings are similar, and almost four times faster than CP-AB algorithm and 15 times faster than the CP-ALS algorithm. It is the same as the results from simulated datasets that the convergence process of CP-AD and CP-AB are more robust than CP-AG indicated by smoother declining residual curves. However, much more oscillations can be observed in the CP-AG results. As we know from the Methods section, CP-AB works in a deterministic way to find the best step size, thus it leads to a robust convergence behavior. The CP-AD algorithm takes into account the recent iterative subspace and the optimization of weights tends to make the extrapolation robust. However, CP-AG (also CP-AO) performs the extrapolation in a heuristic manner that can easily makes the algorithm go too far and most likely lead to a high residual at some steps in real cases.
The final results of all the CP-ALS algorithms on the fluorescence spectroscopy data can be seen from Table 3. It is shown that all the proposed extrapolated algorithms, except for CP-AO, find the same solutions as the non-accelerated CP-ALS algorithm indicated by the same explained fit percentage 99.68%. Moreover, CP-AD is advantageous in significantly decreasing the number of iterations needed for the convergence, taking the fewest number of iterations to converge on the fluorescence data. On the other hand, CP-ALS takes many (1401) iterations to converge in this case.

Real datasets: Hyperspectral data

All the algorithms are further tested on the high dimensional hyperspectral data. The results are shown on Figure 4. Similar to the simulated data results, all the proposed extrapolated algorithms are faster than CP-AB algorithm. For this hyperspectral data, CP-AB and CP-AG are a bit slower than CP-ALS algorithm. CP-AO is the fastest in time consumption being almost three times faster than CP-AB, followed by the CP-AD and CP-AG. The jumps behaviors of CP-AO and CP-AG can be clearly observed from Figure 4, while the residual curve of CP-AD is again smooth being similar to the curve of CP-AB. Regarding the number of iterations, all the extrapolated CP algorithms manage to decrease the needed number of iteration for CP-ALS convergence except for CP-AG. CP-AB and CP-AO algorithms take almost three times less number of iterations than CP-ALS algorithm. The details of number of iterations and explained fit percentages of all the CP-ALS algorithms on this hyperspectral data are shown in Table 3. It can be clearly seen that all the algorithms reach the same explained fit percentage as 99.16%. The difference of consumed number of iterations between CP-AB, CP-AO and CP-AD is minor.

Real dataset: EEG data

The results of the algorithms on EEG data is presented on Figure 5, and shows overall similar patterns as above. As shown by Figure 5, the proposed extrapolated algorithms successfully accelerate the convergence of CP-ALS algorithm on the EEG data. CP-AB is quite slow in this case and it is in fact slower than non-accelerated CP-ALS algorithm. The CP-AO is almost three times faster than CP-AB, which is the fastest. Again, many significant jumps can be observed from the residual curves of CP-AO and CP-AG algorithms while only some minor jumps occur in the convergence process of CP-AD. Thus CP-AD is again more robust than CP-AO and CP-AG on the EEG data. The minor jumps of CP-AD could be speculated to be due to standard version of CP-AD proceeds without checking if the DIIS step actually improves the step, or it could be due to subtle details of the numerics such the threshold settings in the pseudo inverse calculation (set to 1e-12 by default). We performed additional computations, given in Figure S5 in the supporting information, to clarify the situations. It is seem that the results depends somewhat on the pseudo inverse threshold. The convergence curve for the conservative algorithm, CP-ADc, has monotonically decaying residual norm. The fewer iterations (10-20) means that CP-ADc is in fact computationally competitive with the standard CP-AD. Thus, at least in some cases, the additional safety of the CP-ADc may be worth investing, but do note that even in the present case there is almost no gain in speed by CP-ADc despite the improved convergence in terms of iteration number. As shown from Table 3, all the algorithms reach the same explained fit percentage being 80.30%. And all the proposed algorithms significantly decrease the number of iterations needed for the convergence. In particular, CP-AB and CP-AO takes more than three times less number of iterations than CP-ALS.

Conclusions

In this work, we introduce some new alternating least square algorithms for calculating the CP tensor decomposition and accelerating the convergence of the workhorse CP-ALS algorithm. The proposed algorithms are tested on a set of simulated datasets and various real experimental datasets, and a comprehensive comparison has been made with the benchmark extrapolated CP-AB algorithm. The experimental results show that our proposed algorithms are able to successfully and significantly accelerate the CP decomposition whilst maintaining a good quality of solution. For example, CP-AD and CP-AO achieve the fastest convergence in most cases and significantly decrease the needed number of iterations for CP decomposition convergence. In particular, the CP-AD was found to be capable of accelerating the convergence of CP decomposition in a fairly robust way. We also show that the CP-AB algorithm sometimes fails to accelerate CP-ALS. The presented work provides new insights for accelerating CP tensor decomposition as well as the practical evidence for employing extrapolation to improve CP tensor computation. Given the simplicity of implementation, the efficiency and robustness of CP-AD we foresee it will be useful in many different contexts. Moreover, extensions of the proposed methods onto other tensor models such as Tucker 3 model, as well as the constrained tensor models, will be also be very interesting. Further works on extending the proposed CP algorithms onto higher-order tensors data such as fourth-order or fifth-order will also be valuable.

Supporting information

Conservative CP-AD algorithm (CP-ADc, Algorithm 4)
Results of different CP algorithms on balanced size simulated data with rank 6 and 10% noise level (Figure S1)
Results of different CP algorithms on imbalanced size simulated data with rank 6 and 10% noise level (Figure S2)
Results of different CP algorithms on balanced size simulated data with rank 7 and 10% noise level (Figure S3)
Results of different CP algorithms on imbalanced size simulated data with rank 7 and 10% noise level (Figure S4)
Results comparison of the CP-AD algorithms on EEG tensor with different pseudo inverse threshold values in the DIIS extrapolation procedure(Figure S5)

Algorithm 4 CP-ADc algorithm

```
 1: initialization: Do SVD on raw tensor $\underline{\mathbf{X}}$ to get a set of initial factors matrices (A,B,C), and set
    $i = 1$
 2: while $i = 1, 2, 3, \dots$ do
 3:     update the least square factor matrices - first use CP-ALS:
 4:     $\mathbf{A} = \mathbf{X}_{I\times JK}[(\mathbf{C} \odot \mathbf{B})((\mathbf{B}^T\mathbf{B}) \oplus (\mathbf{C}^T\mathbf{C}))^+]$;
 5:     $\mathbf{B} = \mathbf{X}_{J\times IK}[(\mathbf{C} \odot \mathbf{A})((\mathbf{A}^T\mathbf{A}) \oplus (\mathbf{C}^T\mathbf{C}))^+]$;
 6:     $\mathbf{C} = \mathbf{X}_{K\times IJ}[(\mathbf{B} \odot \mathbf{A})((\mathbf{A}^T\mathbf{A}) \oplus (\mathbf{B}^T\mathbf{B}))^+]$;
 7:     calculate the residual tensor $\underline{\mathbf{R}}_i = \underline{\mathbf{X}} - \underline{\mathbf{X}}_i$
 8:     if $i \leq n_{trial}$ then
 9:         store the factor matrices and the stacked residual vector
10:     else
11:         Perform CP-AD acceleration:
12:         store the factor matrices and the stacked residual vector
13:         Use the last $n_{trial}$ residuals to compute the $\mathbf{M}$ matrix,$\mathbf{M}_{ij} = \mathbf{r}_i^T\mathbf{r}_j$
14:         Calculate the weights of the last $n_{trial}$ factor matrix estimates according to equation 9
15:         compute improved factor matrices according to equation 7 to get optimized $(\mathbf{A}_i, \mathbf{B}_i, \mathbf{C}_i)$
16:         calculate the residual tensor $\underline{\mathbf{R}}_i = \underline{\mathbf{X}} - \underline{\mathbf{X}}_i$
17:         evaluation the extrapolation:
18:         if $e_i \leq e_{i-1}$ then
19:             $\mathbf{A} \leftarrow \mathbf{A_i}$
20:             $\mathbf{B} \leftarrow \mathbf{B_i}$
21:             $\mathbf{C} \leftarrow \mathbf{C_i}$
22:         else
23:             $\mathbf{A} \leftarrow \mathbf{A}$
24:             $\mathbf{B} \leftarrow \mathbf{B}$
25:             $\mathbf{C} \leftarrow \mathbf{C}$
26:         end if
27:     end if
28:     set $i = i + 1$
29:     repeat until the residual change reaches the predefined criterion
30: end while
```

Author contributions

Conceptualization: H.Y., O.C. Data curation: H.Y. Formal analysis: H.Y., K.L., O.C. Methodology: H.Y., K.L., O.C. Writing—original draft: H.Y. Writing—review and editing: H.Y., K.L., O.C. All authors have given approval to the final version of the manuscript.

Acknowledgements

We acknowledge the funding support from Villum Foundation in Denmark and the scientific discussion with Dr. Ian H Godtliebsen from Aarhus University.

Availability of data and materials

The codes, data and results are available upon request from the author.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Department of Chemistry, Aarhus University, Aarhus, Denmark. [2]Department of Computer Science, Aarhus University, Aarhus, Denmark.

References

1. Sanou, I.W., Redon, R., Luciani, X., Mounier, S.: Online nonnegative and sparse canonical polyadic decomposition of fluorescence tensors. Chemometrics and Intelligent Laboratory Systems **225**, 104550 (2022)
2. Yan, X.-F., Liang, Y.-M., Zhou, B., Bin, J., Kang, C.: Enhancing the selectivity of liquid chromatography–mass spectrometry by using trilinear decomposition on lc-ms data: An application to three-way calibration of coeluting analytes in human plasma. Journal of Separation Science **43**(13), 2718–2727 (2020)
3. Sun, W., Braatz, R.D.: Opportunities in tensorial data analytics for chemical and biological manufacturing processes. Computers & Chemical Engineering **143**, 107099 (2020)
4. Schütt, K.T., Arbabzadah, F., Chmiela, S., Müller, K.R., Tkatchenko, A.: Quantum-chemical insights from deep tensor neural networks. Nature communications **8**(1), 13890 (2017)

5. Trinklein, T.J., Cain, C.N., Ochoa, G.S., Schöneich, S., Mikaliunaite, L., Synovec, R.E.: Recent advances in gc× gc and chemometrics to address emerging challenges in nontargeted analysis. Analytical Chemistry **95**(1), 264–286 (2023)
6. Guo, Y., Wang, X., Lan, X., Su, T.: Traffic target location estimation based on tensor decomposition in intelligent transportation system. IEEE Transactions on Intelligent Transportation Systems (2022)
7. Taguchi, Y., Turki, T.: Adapted tensor decomposition and pca based unsupervised feature extraction select more biologically reasonable differentially expressed genes than conventional methods. Scientific Reports **12**(1), 17438 (2022)
8. Zhang, R., Cheng, L., Wang, S., Lou, Y., Wu, W., Ng, D.W.K.: Tensor decomposition-based channel estimation for hybrid mmwave massive mimo in high-mobility scenarios. IEEE Transactions on Communications **70**(9), 6325–6340 (2022)
9. Kronik, O.M., Liang, X., Nielsen, N.J., Christensen, J.H., Tomasi, G.: Obtaining clean and informative mass spectra from complex chromatographic and high-resolution all-ions-fragmentation data by nonnegative parallel factor analysis 2. Journal of Chromatography A **1682**, 463501 (2022)
10. Wells, M.J., Hooper, J., Mullins, G.A., Bell, K.Y.: Development of a fluorescence eem-parafac model for potable water reuse monitoring: Implications for inter-component protein–fulvic–humic interactions. Science of the Total Environment **820**, 153070 (2022)
11. Skantze, V., Wallman, M., Sandberg, A.-S., Landberg, R., Jirstrand, M., Brunius, C.: Identification of metabotypes in complex biological data using tensor decomposition. Chemometrics and Intelligent Laboratory Systems **233**, 104733 (2023)
12. Madsen, N.K., Jensen, R.B., Christiansen, O.: Calculating vibrational excitation energies using tensor-decomposed vibrational coupled-cluster response theory. The Journal of Chemical Physics **154**(5), 054113 (2021)
13. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM review **51**(3), 455–500 (2009)
14. Harshman, R.: Foundations of the parafac procedure: Model and conditions for an explanatory factor analysis. Technical Report UCLA Working Papers in Phonetics 16 (1970)
15. Carroll, J.D., Chang, J.-J.: Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. Psychometrika **35**(3), 283–319 (1970)
16. Tomasi, G., Bro, R.: A comparison of algorithms for fitting the parafac model. Computational Statistics & Data Analysis **50**(7), 1700–1734 (2006)
17. Phan, A.-H., Tichavskỳ, P., Cichocki, A.: Partitioned hierarchical alternating least squares algorithm for cp tensor decomposition. In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2542–2546 (2017). IEEE
18. Battaglino, C., Ballard, G., Kolda, T.G.: A practical randomized cp tensor decomposition. SIAM Journal on Matrix Analysis and Applications **39**(2), 876–901 (2018)
19. Ranadive, T.M., Baskaran, M.M.: Large-scale sparse tensor decomposition using a damped gauss-newton method. In: 2020 IEEE High Performance Extreme Computing Conference (HPEC), pp. 1–8 (2020). IEEE
20. Wang, D., Cong, F.: An inexact alternating proximal gradient algorithm for nonnegative cp tensor decomposition. Science China Technological Sciences **64**(9), 1893–1906 (2021)
21. Abed-Meraim, K., Trung, N.L., Hafiane, A., *et al.*: A fast randomized adaptive cp decomposition for streaming tensors. In: ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 2910–2914 (2021). IEEE
22. Phan, A.-H., Tichavskỳ, P., Cichocki, A.: Error preserving correction: A method for cp decomposition at a target error bound. IEEE Transactions on Signal Processing **67**(5), 1175–1190 (2018)
23. Sidiropoulos, N.D., Bro, R.: On the uniqueness of multilinear decomposition of n-way arrays. Journal of Chemometrics: A Journal of the Chemometrics Society **14**(3), 229–239 (2000)
24. Bro, R.: Multi-way analysis in the food industry. Models, Algorithms, and Applications. PhD thesis. (1998)
25. Yu, H., Augustijn, D., Bro, R.: Accelerating parafac2 algorithms for non-negative complex tensor decomposition. Chemometrics and Intelligent Laboratory Systems **214**, 104312 (2021)
26. Tian, K., Wu, L., Min, S., Bro, R.: Geometric search: A new approach for fitting parafac2 models on gc-ms data. Talanta **185**, 378–386 (2018)
27. Helgaker, T., Jørgensen, P., Olsen, J.: Molecular Electronic-Structure Theory. Wiley, Chichester ; New York (2000)
28. Pulay, P.: Convergence acceleration of iterative sequences. the case of scf iteration. Chem. Phys. Lett. **73**(2), 393–398 (1980). doi:10.1016/0009-2614(80)80396-4
29. Ettenhuber, P., Jørgensen, P.: Discarding information from previous iterations in an optimal way to solve the coupled cluster amplitude equations. Journal of Chemical Theory and Computation **11**(4), 1518–1524 (2015)
30. Nesterov, Y.E.: A method of solving a convex programming problem with convergence rate o\bigl(k^2\bigr). In: Doklady Akademii Nauk, vol. 269, pp. 543–547 (1983). Russian Academy of Sciences
31. Ang, A.M.S., Gillis, N.: Accelerating nonnegative matrix factorization algorithms using extrapolation. Neural computation **31**(2), 417–439 (2019)
32. Ang, A., Cohen, J.E., Gillis, N.: Accelerating approximate nonnegative canonical polyadic decomposition using extrapolation. In: GRETSI 2019-XXVIIème Colloque Francophone de Traitement du Signal et des Images, pp. 1–4 (2019)
33. Lawaetz, A.J., Bro, R., Kamstrup-Nielsen, M., Christensen, I.J., Jørgensen, L.N., Nielsen, H.J.: Fluorescence spectroscopy as a potential metabonomic tool for early detection of colorectal cancer. Metabolomics **8**, 111–121 (2012)
34. Ding, M., Fu, X., Huang, T.-Z., Wang, J., Zhao, X.-L.: Hyperspectral super-resolution via interpretable block-term tensor modeling. IEEE Journal of Selected Topics in Signal Processing **15**(3), 641–656 (2020)
35. Hyperspectral image datasets source. https://ehu.eus/ccwintco/index.php?title=GIC-experimental-databases. Accessed: 2023-12-13
36. Yu, H., Bro, R.: Parafac2 and local minima. Chemometrics and Intelligent Laboratory Systems **219**, 104446

(2021)
37. Mørup, M., Hansen, L.K., Arnfred, S.M.: Erpwavelab: A toolbox for multi-channel analysis of time–frequency transformed event related potentials. Journal of neuroscience methods **161**(2), 361–368 (2007)
38. EEG datasets source. http://www.erpwavelab.org/index_files/Page361.htm. Accessed: 2023-12-13
39. Lorenzo-Seva, U., Ten Berge, J.M.: Tucker's congruence coefficient as a meaningful index of factor similarity. Methodology **2**(2), 57–64 (2006)

Table 1 Results of various CP-ALS algorithms on simulated tensors with balanced size (150*150*150) and different number of ranks.

Algorithms	Ranks	Noise level	T(A)*	T(B)	T(C)	Time (S)**	NI***
CP-ALS	5	0.1	1.00	1.00	1.00	12	150
	5	0.01	1.00	1.00	1.00	12	145
	5	0.001	1.00	1.00	1.00	12	152
	6	0.1	1.00	1.00	1.00	13	166
	6	0.01	1.00	1.00	1.00	13	168
	6	0.001	1.00	1.00	1.00	14	170
	7	0.1	1.00	1.00	1.00	14	174
	7	0.01	1.00	1.00	1.00	15	183
	7	0.001	1.00	1.00	1.00	19	178
CP-AB	5	0.1	1.00	1.00	1.00	15	39
	5	0.01	1.00	1.00	1.00	19	46
	5	0.001	1.00	1.00	1.00	17	42
	6	0.1	1.00	1.00	1.00	17	45
	6	0.01	1.00	1.00	1.00	19	52
	6	0.001	1.00	1.00	1.00	17	44
	7	0.1	1.00	1.00	1.00	17	45
	7	0.01	1.00	1.00	1.00	18	43
	7	0.001	1.00	1.00	1.00	24	47
CP-AD	5	0.1	1.00	1.00	1.00	5	28
	5	0.01	1.00	1.00	1.00	5	29
	5	0.001	1.00	1.00	1.00	4	26
	6	0.1	1.00	1.00	1.00	4	28
	6	0.01	1.00	1.00	1.00	5	31
	6	0.001	1.00	1.00	1.00	4	28
	7	0.1	1.00	1.00	1.00	5	30
	7	0.01	1.00	1.00	1.00	5	30
	7	0.001	1.00	1.00	1.00	6	33
CP-AG	5	0.1	1.00	1.00	1.00	7	69
	5	0.01	1.00	1.00	1.00	7	67
	5	0.001	1.00	1.00	1.00	7	70
	6	0.1	1.00	1.00	1.00	7	68
	6	0.01	1.00	1.00	1.00	7	75
	6	0.001	1.00	1.00	1.00	8	82
	7	0.1	1.00	1.00	1.00	8	87
	7	0.01	1.00	1.00	1.00	8	79
	7	0.001	1.00	1.00	1.00	10	86
CP-AO	5	0.1	1.00	1.00	1.00	4	42
	5	0.01	1.00	1.00	1.00	4	40
	5	0.001	1.00	1.00	1.00	4	43
	6	0.1	1.00	1.00	1.00	4	48
	6	0.01	1.00	1.00	1.00	4	46
	6	0.001	1.00	1.00	1.00	4	47
	7	0.1	1.00	1.00	1.00	4	48
	7	0.01	1.00	1.00	1.00	4	52
	7	0.001	1.00	1.00	1.00	6	53

*T(A): Average tucker congruence coefficient between decomposed factor matrix and real factor matrix for 10 tensor datasets, T(B) and T(C) are defined in the same way.
**Time: Average time consumption of the algorithm on 10 tensor datasets
***NI: Average number of iterations the algorithm takes to converge on 10 tensor datasets.

Table 2 Results of various CP-ALS algorithms on simulated tensors with imbalanced size (250*1000*10) and different number of ranks.

Algorithms	Ranks	Noise level	T(A)*	T(B)	T(C)	Time (S)**	NI***
CP-ALS	5	0.1	1.00	1.00	1.00	13	172
	5	0.01	1.00	1.00	1.00	12	158
	5	0.001	1.00	1.00	1.00	14	179
	6	0.1	1.00	1.00	1.00	16	195
	6	0.01	1.00	1.00	1.00	16	207
	6	0.001	1.00	1.00	1.00	13	168
	7	0.1	1.00	1.00	1.00	17	207
	7	0.01	1.00	1.00	1.00	19	229
	7	0.001	1.00	1.00	1.00	19	231
CP-AB	5	0.1	1.00	1.00	1.00	16	50
	5	0.01	1.00	1.00	1.00	14	46
	5	0.001	1.00	1.00	1.00	16	50
	6	0.1	1.00	1.00	1.00	19	59
	6	0.01	1.00	1.00	1.00	18	57
	6	0.001	1.00	1.00	1.00	16	52
	7	0.1	1.00	1.00	1.00	18	58
	7	0.01	1.00	1.00	1.00	21	67
	7	0.001	1.00	1.00	1.00	21	67
CP-AD	5	0.1	1.00	1.00	1.00	5	31
	5	0.01	1.00	1.00	1.00	5	32
	5	0.001	1.00	1.00	1.00	5	31
	6	0.1	1.00	1.00	1.00	6	43
	6	0.01	1.00	1.00	1.00	7	47
	6	0.001	1.00	1.00	1.00	6	42
	7	0.1	1.00	1.00	1.00	7	45
	7	0.01	1.00	1.00	1.00	8	49
	7	0.001	1.00	1.00	1.00	7	52
CP-AG	5	0.1	1.00	1.00	1.00	8	80
	5	0.01	1.00	1.00	1.00	7	81
	5	0.001	1.00	1.00	1.00	7	81
	6	0.1	1.00	1.00	1.00	9	92
	6	0.01	1.00	1.00	1.00	9	94
	6	0.001	1.00	1.00	1.00	8	86
	7	0.1	1.00	1.00	1.00	9	89
	7	0.01	1.00	1.00	1.00	11	110
	7	0.001	1.00	1.00	1.00	10	109
CP-AO	5	0.1	1.00	1.00	1.00	4	47
	5	0.01	1.00	1.00	1.00	4	44
	5	0.001	1.00	1.00	1.00	4	49
	6	0.1	1.00	1.00	1.00	5	52
	6	0.01	1.00	1.00	1.00	5	52
	6	0.001	1.00	1.00	1.00	4	45
	7	0.1	1.00	1.00	1.00	5	53
	7	0.01	1.00	1.00	1.00	5	52
	7	0.001	1.00	1.00	1.00	6	61

*T(A): Average tucker congruence coefficient between decomposed factor matrix and real factor matrix for 10 tensor datasets, T(B) and T(C) are defined in the same way.
**Time: Average time consumption of the algorithm on 10 tensor datasets
***NI: Average number of iterations the algorithm takes to converge on 10 tensor datasets.

Table 3 Results of various CP-ALS algorithms on real chemical tensors—fluorescence data, hyperspectral data and EEG data.

Algorithms	Fluorescence data		Hyperspectral data		EEG data	
	EFP (%)*	NI**	EFP (%)*	NI**	EFP (%)*	NI**
CP-ALS	99.68	1401	99.16	57	80.30	145
CP-AB	99.68	70	99.16	22	80.30	44
CP-AD	99.68	52	99.16	30	80.30	76
CP-AG	99.68	102	99.16	59	80.30	72
CP-AO	99.67***	1084	99.16	24	80.30	46

*EFP: Explained fit percentages of the algorithm.
**NI: Number of iterations the algorithm takes to converge.
***99.67: Local minima solution.