EXTRAPOLATED ALTERNATING ALGORITHMS FOR APPROXIMATE CANONICAL POLYADIC DECOMPOSITION

Andersen Man Shun Ang†, *Jeremy E. Cohen*‡, *Le Thi Khanh Hien*†, *Nicolas Gillis*†

†Department of Mathematics and Operational Research, Université de Mons, Belgium
‡CNRS, Université de Rennes, Inria, IRISA Campus de Beaulieu, Rennes, France

ABSTRACT

Tensor decompositions have become a central tool in machine learning to extract interpretable patterns from multiway arrays of data. However, computing the approximate Canonical Polyadic Decomposition (aCPD), one of the most important tensor decomposition model, remains a challenge. In this work, we propose several algorithms based on extrapolation that improve over existing alternating methods for aCPD. We show on several simulated and real data sets that carefully designed extrapolation can significantly improve the convergence speed hence reduce the computational time, especially in difficult scenarios.

***Index Terms*—** Canonical Polyadic Decomposition, Tensor, Non-convex Optimization, Block-coordinate Descent, Acceleration

1. PROBLEM STATEMENT

Let $\otimes$ be the tensor product $[a^{(1)} \otimes \ldots \otimes a^{(p)}]_{i_1 \ldots i_p} = \prod_{j=1}^{p} a^{(j)}_{i_j}$ with $a^{(j)} \in \mathbb{R}^{n_j}$ and $p \in \mathbb{N}^*$, set $\mathbf{n} = \times_j n_j$ for a collection of $n_j \in \mathbb{N}^*$. We are interested in solving efficiently the following approximate Canonical Polyadic Decomposition (aCPD) optimization problem:

Definition 1 (aCPD) *Given a tensor $T \in \mathbb{R}^{\mathbf{n}}$ of order p and an integer r, find a tensor $\hat{T}$ such that*

$$\hat{T} = \underset{\operatorname{rank}(G) \leq r}{\operatorname{argmin}} \|T - G\|_F^2, \tag{1}$$

where the rank of a tensor G is defined as

$$\min \left\{ r \in \mathbb{N} \mid \exists a_i^{(j)} \in \mathbb{R}^{n_j},\ G = \sum_{i=1}^{r} \bigotimes_{j=1}^{p} a_i^{(j)} \right\}. \tag{2}$$

The aCPD problem is ill-posed in the sense that a solution might not exist, since the set of rank r tensors is not closed as soon as $r > 1$ and $p > 2$ [1]. This poses serious problems in practice. It has been documented that the estimates of each block $A^{(j)} = [a_1^{(j)}, \ldots, a_r^{(j)}]$ ($1 \leq j \leq p$) may in consequence have pairs of columns growing to infinity while canceling each-other out, even if a best rank r approximation exists; see [2] and the references therein. This degeneracy causes "swamps" in the cost function decrease, such as shown in Figure 1. The cost function is also non-convex albeit quadratic with respect to each block $A^{(j)}$ ($1 \leq j \leq p$). Therefore, computing aCPD remains a challenging task in general. We refer the interested reader to [3] for a comprehensive survey on these questions.

AMSA, LTKH and NG acknowledge the support by the European Research Council (ERC starting grant No 679515), and by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlanderen (FWO) under EOS project O005318F-RG47.

2. SOME EXISTING ALTERNATING ALGORITHMS

As tensor decompositions have become an important and extensively studied topic in data science, it is out of the scope of this paper to summarize all the literature on how to compute aCPD. Therefore, in what follows, we focus on alternating methods, which update one block of variables $A^{(j)}$ at a time while keeping the others fixed.

There are mainly two categories of alternating algorithms to compute aCPD: Exact Block-Coordinate Descent (EBCD) algorithms, and Approximate BCD (ABCD) algorithms. EBCD algorithms feature block-wise optimal updates. The most well-known EBCD algorithm for aCPD is Alternating Least Squares (ALS), also called CP-ALS or sometimes PARAFAC (which is also another name for aCPD). ALS sequentially updates the blocks $A^{(j)}$ as follows:

$$\hat{A}^{(j)} = g(T, A^{(l \neq j)}) := T_{[j]} {B^{(j)}}^{\dagger}, \tag{3}$$

where $T_{[j]}$ is the j-th unfolding of T, as defined for instance in [4], $\odot$ is the Khatri-Rao product and ${B^{(j)}}^T = \bigodot_{\substack{l=p \\ l \neq j}}^{1} A^{(l)}$. ALS has no convergence guarantee since each block update may have more than one solution [5]. There are some local convergence guarantee though [6]. Another simple BCD algorithm is Hierarchical ALS (HALS), which updates each column of each $A^{(j)}$ sequentially. While HALS has scarcely been used for solving aCPD (contrary to its nonnegative counterpart [7]), it may in principle be faster than ALS for large tensors since no linear systems are to be solved at each iteration due to the separability of the objective function. A variant of HALS, A-HALS, updates the columns of each factor $A^{(j)}$ with several cycles before jumping to another factor. This allows to reduce the computational cost as some matrix products can be reused [8].

ABCD algorithms, such as alternating gradient methods, do not solve each subproblem optimally. These methods have scarcely been considered for solving aCPD, in favor of all-at-once gradient-based approaches [9].

3. PROPOSED APPROACHES: IBPG AND HERBCD

In the following, we introduce two algorithms for computing aCPD that make use of extrapolation in two different ways. Extrapolation for escaping saddle points and enhancing convergence speed is an intensively studied topic in machine learning, in particular in the context of deep learning where the cost functions to minimize are highly non-convex and gradient-based algorithms might require a "push" to escape bad regions of the search space faster. Our goal is to show empirically that extrapolation, on top of enhancing empirical convergence speed in difficult cases, also helps escaping "swamps" when computing aCPD. This observation is actually not new, and may be traced back to seminal work by Harshman [10]. We provide a fresh view on these issues by using more recent optimization techniques.

For convenience, let us define the cost function F as

$$F(A^{(1)},\dots,A^{(p)}) = \|T - \sum_{i=1}^{r}\bigotimes_{j=1}^{p} a_i^{(j)}\|_F^2. \quad (4)$$

3.1. Inertial Block Proximal Gradient (iBPG)

A recent trend in non-convex optimization for machine learning, which takes root in the seminal work by Nesterov [11], is to make use of extrapolation of the iterates to enhance convergence speed of the cost function. Although such techniques were originally designed for convex smooth problems and gradient descent, extrapolation has been investigated in the context of non-smooth [12], non-convex [13] problem and in conjunction with alternating gradient descent [14]. It has also been shown that extrapolation of the iterates can be interpreted in the realm of harmonic mechanical systems, shedding light on the performance enhancement [15, 16]. These techniques have been seldom used for solving aCPD, despite some recent attempts [17, 18] summarized in Section 5.

Such an alternating (proximal) gradient descent algorithm with extrapolation was recently proposed [19], which we refer to as iBPG, to solve a general nonconvex nonsmooth block separable composite optimization problem. iBPG embraces some advanced features of

Algorithm 1 iBPG for CPD

```
1: Initialization: Choose $\delta_w = 0.99$, $\beta = 1.01$, $t_0 = 1$, and 2 sets of initial factor matrices $\left(A_{-1}^{(1)},\dots,A_{-1}^{(p)}\right)$ and $\left(A_0^{(1)},\dots,A_0^{(p)}\right)$. Set $k = 1$.
2: Set $A_{\text{prev}}^{(j)} = A_{-1}^{(j)}, j = 1,\dots,p$.
3: Set $A_{\text{cur}}^{(j)} = A_0^{(j)}, j = 1,\dots,p$.
4: repeat
5:     for $j = 1,\dots,p$ do
6:         $t_k = \frac{1}{2}(1+\sqrt{1+4t_{k-1}^2})$, $\hat{w}_{k-1} = \frac{t_{k-1}-1}{t_k}$
7:         $w_{k-1}^{(j)} = \min\left(\hat{w}_{k-1}, \delta_w\sqrt{\frac{L_{k-2}^{(j)}}{L_{k-1}^{(j)}}}\right)$,
8:         $L_k^{(j)} = \left\|\left(B_k^{(j)}\right)^T B_k^{(j)}\right\|$
9:         repeat
10:            Compute two extrapolation points
                   $\hat{A}^{(j,1)} = A_{\text{cur}}^{(j)} + w_{k-1}^{(j)}\left(A_{\text{cur}}^{(j)} - A_{\text{prev}}^{(j)}\right)$,
                   $\hat{A}^{(j,2)} = A_{\text{cur}}^{(j)} + \beta w_{k-1}^{(j)}\left(A_{\text{cur}}^{(j)} - A_{\text{prev}}^{(j)}\right)$
11:            Set $A_{\text{prev}}^{(j)} = A_{\text{cur}}^{(j)}$.
12:            Update $A_{\text{cur}}^{(j)}$ by gradient step:
                   $A_{\text{cur}}^{(j)} = \hat{A}^{(j,2)} - \frac{1}{L_{k-1}^{(j)}}\left(\hat{A}^{(j,1)}\left(B_{k-1}^{(j)}\right)^T - \mathcal{T}_{[j]}\right)B_{k-1}^{(j)}$.
13:        until some criteria is satisfied
14:        Set $A_k^{(j)} = A_{\text{cur}}^{(j)}$.
15:    end for
16:    Set $k = k + 1$.
17: until some criteria is satisfied
```

acceleration methods using extrapolation:

- iBPG uses two different extrapolation points to evaluate the gradient and to add inertial force. This feature was experimentally shown to improve convergence compared to the use of a single extrapolation point.
- iBPG does not require a restarting step: convergence is guaranteed without any restart. This is in contract with most algorithms using extrapolation in the non-convex case where a restarting step is necessary to ensure convergence [17, 14]: a step is accepted only if the objective function decreases and, when this is not the case, the algorithm restarts by taking a standard gradient step. This feature is very useful when evaluating the objective function is expensive.
- iBPG is very flexible in the choice of the order in which the blocks are updated: for example each matrix factor can be updated several times allowing to reuse some computations (like in A-HALS [8]) leading to more updates at a lower computational cost.

iBPG is proved to have sub-sequential convergence under some mild conditions, and global convergence under some additional assumptions. iBPG can easily be instantiated for aCPD, the resulting algorithm is summarized in Algorithm 3.1. As the choice of the parameters in Algorithm 3.1 satisfies the relaxed conditions in [19, Remark 4.7], we can derive from [19, Theorem 4.8] that iBPG for solving aCPD is guaranteed to have (at least) a sub-sequential convergence; see [19, Section 5] for a similar explanation in the case of non-negative matrix factorization problem.

3.2. Heuristic Extrapolation and Restart (her) BCD

Although alternating gradient-based approaches are not state-of-the-art at the moment for computing aCPD, BCD algorithms on the other hand are extremely popular, in particular the ALS algorithm, mostly due to its simplicity and efficiency for simple problems. However, ALS is known to converge slowly for instance when the factors $A^{(j)}$ are ill-conditioned. In this paper, we introduce an extrapolation of the factor estimates **between** each block update. Moreover, the extrapolation technique is not a straightforward heuristic line search such as described in [20, pp.95–96], but mimics Nesterov's extrapolation by introducing pairing variables $Z^{(j)}$. For instance, when updating the jth factor in the ALS algorithm at the kth iteration, the update is modified as:

$$A_k^{(j)} = g\left(T, \left[Z_k^{(l<j)}, Z_{k-1}^{(l>j)}\right]\right) \text{ as defined in (3)} \quad (5)$$

$$Z_k^{(j)} = A_k^{(j)} + \beta_k\left(A_k^{(j)} - A_{k-1}^{(j)}\right), \quad (6)$$

where β_k is updated heuristically (see Algorithm 3.2) using three parameters $(\eta, \bar{\gamma}, \gamma)$ following the strategy described in [21]. In short, the idea is to use a restart criterion $\hat{F}_k = F(A_k^{(p)}; Z_k^{(l\neq p)})$, which is the cost evaluated at the pairing variable for the first $p-1$ modes and the original variable at the last mode, to update β_k. When $\hat{F}$ decreases, we grow β (multiply it by a constant $\gamma \geq 1$). When $\hat{F}$ increases, we decrease β (divide it by a constant $\eta \geq 1$). In Algorithm 3.2, $\bar{\beta}$ is the upper bound of β, which is also updated dynamically. Beside updating β, restart is carried out based on $\hat{F}$ to decide whether to keep $A^{(j)}$ or $Z^{(j)}$ in the next iteration. We refer to this procedure as **H**euristic **E**xtrapolation and **R**estart (her). It could be used in the same way to design a her-HALS algorithm and a her-Gradient algorithm, which we do not discuss here do to the space restriction. In fact, any BCD algorithm can be accelerated using her, by extrapolating the partial estimates for each block update. We label such a generic approach herBCD.

Algorithm 2 herALS for CPD

```
1:  Initialization: Choose $\beta_0 \in (0,1)$, $\eta \geq \gamma \geq \bar{\gamma} \geq 1$, and 2 sets of initial factor matrices $(A_0^{(1)}, \dots, A_0^{(p)})$ and $(Z_0^{(1)}, \dots, Z_0^{(p)})$. Set $\bar{\beta}_0 = 1$ and $k = 1$.
2:  repeat
3:      for $j = 1, \dots, p$ do
4:          Update: get $A_k^{(j)}$ as (5).
5:          Extrapolate: get $Z_k^{(j)}$ as (6).
6:      end for
7:      Compute $\hat{F}_k = F(A_k^{(p)}; Z_k^{(l \neq p)})$.
8:      if $\hat{F}_k > \hat{F}_{k-1}$ for $k \geq 2$ then
            Set $Z_k^{(j)} = A_k^{(j)}$ for $j = 1, \dots, p$
            Set $\bar{\beta}_k = \beta_{k-1}$, $\beta_k = \beta_{k-1}/\eta$
9:      else
            Set $A_k^{(j)} = Z_k^{(j)}$ for $j = 1, \dots, p$
            Set $\bar{\beta}_k = \max\{1, \bar{\beta}_{k-1}\bar{\gamma}\}$, $\beta_k = \max\{\bar{\beta}_{k-1}, \beta_{k-1}\}$
10:     end if
11:     Set $k = k + 1$.
12: until some criteria is satisfied
```

3.3. Additional cost of extrapolation

A natural question that arises when modifying well-known algorithms to enhance their convergence speed is the impact of such modifications on the computational time. Indeed, it is often possible to improve the relative decrease of the objective at each iteration with respect to ALS, but doing so while keep a fixed cost per iteration is more challenging.

The cost of iBPG is essentially that of an alternating gradient method. Indeed, the computation of the extrapolation points is negligible given that r is small, since computing the operator norm of an $r \times r$ matrix is cheap (lines 7 and 8 in Algorithm 3.1). Moreover, since there is no need for restart like in herALS, the cost function does not need to be computed at each iteration. Therefore, the cost for each block update boils down to the cost of computing the gradient, which cost is itself known to be dominated by the so-called Matricized Tensor Times Khatri-Rao Product (MTTKRP). The MTTKRP can be efficiently implemented, see for instance [22, 23]. In summary, for small r, one block update of iBPG has essentially the same cost as one block update of ALS since the matrix to inverse in ALS is of size $r \times r$.

The cost of one iteration of herALS is also essentially the same as one iteration of ALS, although this requires a twist on the restart condition. Indeed, to perform restart, it is in theory necessary to check if the cost function is increasing after extrapolation. However, computing the cost function is expensive for aCPD unless the previously computed MTTKRPs can be reused. Note that the restart in herALS is based on the cost evaluated at the pairing variables, for which MTTKRPs have indeed been computed in the ALS update. Although this is not a standard way to perform restart, this allows to keep computational cost low while showing no practical difference.

4. EXPERIMENTS

In this section we compare iBPG and herALS to three algorithms: the original un-accelerated ALS (ALS), the accelerated ALS using Bro's acceleration (Bro-ALS), and the LS-ALS: an accelerated ALS where extrapolation sequence is computed by line search in the style of Anderson Acceleration. See section 5 for description of Bro-ALS and LS-ALS.

In each experiment, the notation $[I, J, K, r]$ denotes the sizes of the tensor (I, J, K), and the factorization rank r. All experiments are run over 20 random initilaziations, and we plot the median of the cost value over these 20 trials. There are two important things to note: all herBCD across all experiments use the same set of default parameters $[\beta_0, \gamma, \bar{\gamma}, \eta] = [0.5, 1.05, 1.01, 1.5]$. All the y-axis of the plots is in the form of $F - F_{\min}$, where F is the cost evaluated at all $A^{(j)}$ and $F_{\min}$ is the minimal possible cost obtained in the experiment across all initializations and algorithms. All the experiments are run with MATLAB (v.2015a) on a laptop with 2.4GHz CPU and 16GB RAM. The codes are available from `https://angms.science/research.html`.

4.1. Synthetic data sets

Figure 1 shows the result over two experiments and details the chosen dimensions of the problem. In both balanced and unbalanced cases that were tested, the data tensor is generated as $\sum_{q=1}^{r} a_q^{(1)} \otimes a_q^{(2)} \otimes a_q^{(3)} + N$, where the ground truth factors $A^{(j)}$ are sampled from a Gaussian distribution with zero mean and unitary variance. Note that we adjust the condition number of $A^{(j)}$ to 100 using the SVD and replacing the singular values by logaritmihc scaled values between 1 and 100. The tensor N is an additive Gaussian noise with zero mean and variance 0.001. The results show that iBPG and herALS are the best algorithm among the five tested algorithms, and in particular seem to avoid the swamp in which ALS lands in the unbalanced case. LS-ALS, which converges fast in terms of iterations, suffers from higher per-iteration cost.

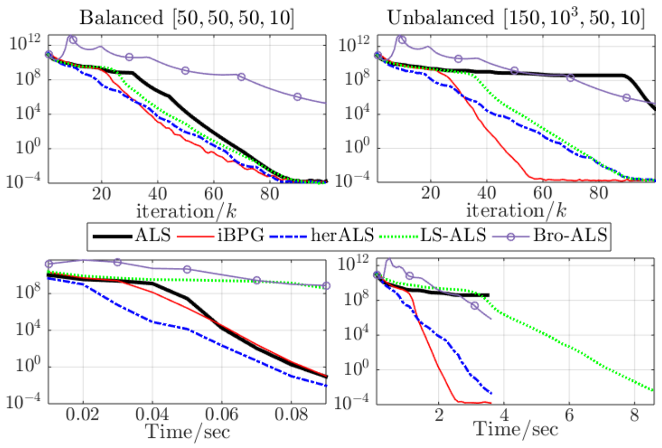

Fig. 1. Median error over 20 runs on synthetic data sets plotted against iterations (top) and time (bottom). On the left is a square tensor $[50, 50, 50, 10]$, and on the right is an unbalanced tensor $[150, 10^3, 50, 10]$. For the unbalanced case, ALS improves very slowly up to the 90th iteration: This phenomenon is often referred to as a swamp in the literature. The proposed extrapolated algorithms do not encounter this issue in this experiment.

4.2. Real data sets

We now show the results on real data sets: Wine[1] (Fig. 2), Hyper spectral data of Indian Pine[2] and Blood plasma[3] (Fig. 4). Again the curves are the median over 20 initializations. All sub-figures in a figure share the same y-axis. Minimal pre-processing is carried out: NaN values (if any) are replaced with zeros.

We observe that herALS performs the best, followed by Bro-ALS. iBPG does not perform as well as for the synthetic data sets.

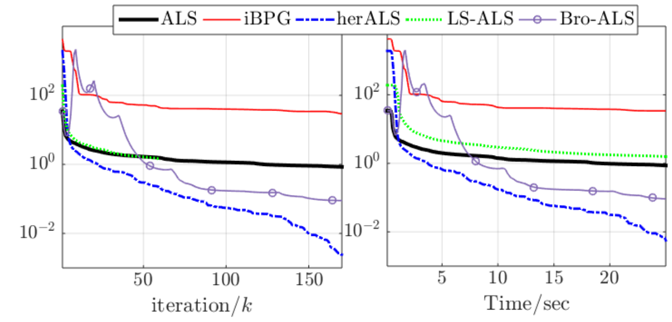

Fig. 2. Results on Wine $[44, 2700, 200, 15]$. iBPG gets stuck on local minima.

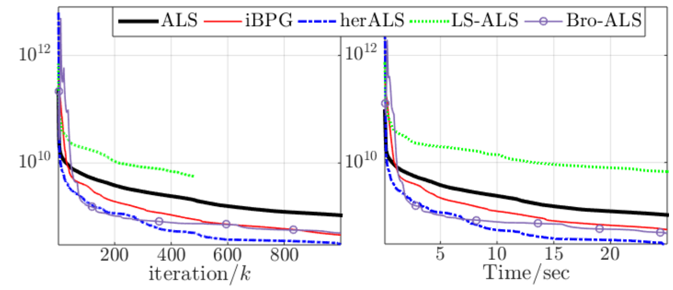

Fig. 3. Results on Indian Pines $[145, 145, 200, 16]$.

5. RELATED PRIOR WORK

Although the idea of extrapolation is not new for ALS [10], there has not been many works tackling extrapolation for speeding up BCD algorithms for aCPD. We are aware of two such works. Bro et al. [20] extrapolate directly the estimated factor using a heuristic approach recalled in [24] which we show can be slower than ALS, although it prevents any appearance of "swamps" in our experiments. Mitchell *et. al.* [18] have proposed a similar extrapolation strategy, where they extrapolate all blocks simultaneously using a shared parameter β_k. In contrast, in this work, we followed the same scheme to compute β_k but the extrapolation is carried out on each block right after the least-squares update, rather than after all least-squares updates. This makes the two approaches quite different. Furthermore an expensive line search (a least square problem) has to be performed to compute the extrapolation weight β. The per-iteration cost is higher than all the other methods in figure 1.

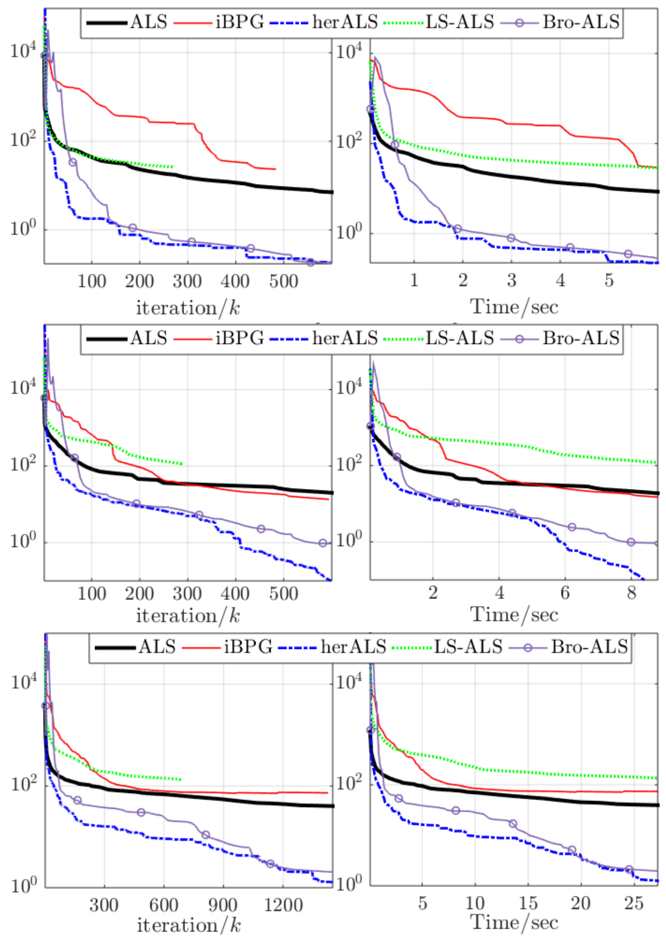

Fig. 4. Results on Blood $[289, 301, 41]$ with $r = 3$ (top), $r = 6$ (mid) and $r = 10$ (bottom). Note that the data has many NaN (data polluted due to Rayleigh scattering), all NaN are replaced by 0. There are therefore many zero fibers in the tensor after such correction.

6. CONCLUSION AND PERSPECTIVES

We have introduced extrapolation-based alternating algorithms for solving aCPD. On a limited set of synthetic experiments with ill-conditioned tensors, the recently proposed iBPG algorithm, which is alternating gradient-based, outperforms workhorse block-coordinate descent algorithm such as ALS, and helps escaping "swamps". The algorithm proposed in this paper, herALS, is a variant of ALS in which iterates are extrapolated, and also performs well without a fine tuning of the hyperparameters. On a few real data sets stemming from fluorescence spectroscopy and remote sensing, herALS outperform all tested methods while iBPG shows mitigated results. Further tests and comparison should therefore be performed to further assess the performance of both iBPG and herALS in specific practical cases. Finally, this work provides further practical evidence that extrapolation helps escaping swamps when computing aCPD.

[1]See `http://www.models.life.ku.dk/Wine_GCMS_FTIR` for data description.

[2]`http://www.ehu.eus/ccwintco/index.php?title=Hyperspectral_Remote_Sensing_Scenes`

[3]See `http://www.models.life.ku.dk/anders-cancer`

7. REFERENCES

[1] V. De Silva and L.-H. Lim, "Tensor rank and the ill-posedness of the best low-rank approximation problem," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 3, pp. 1084–1127, 2008.

[2] P. Comon, X. Luciani, and A. L.F. De Almeida, "Tensor decompositions, alternating least squares and other tales," *Journal of Chemometrics*, vol. 23, no. 7-8, pp. 393–405, 2009.

[3] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, "Tensor decomposition for signal processing and machine learning," *IEEE Transactions on Signal Processing*, vol. 65, no. 13, pp. 3551–3582, 2017.

[4] T. G. Kolda and B. W. Bader, "Tensor decompositions and applications," *SIAM Review*, vol. 51, no. 3, pp. 455–500, sep 2009.

[5] D. P. Bertsekas, *Nonlinear programming*, Athena scientific Belmont, 1999.

[6] A. Uschmajew, "Local convergence of the alternating least squares algorithm for canonical tensor approximation," *SIAM J. matrix Analysis*, vol. 33, no. 2, pp. 639–652, 2012.

[7] A. Cichocki, R. Zdunek, A. H. Phan, and S-I. Amari, *Nonnegative Matrix and Tensor Factorization*, Wiley, 2009.

[8] N. Gillis and F. Glineur, "Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization," *Neural computation*, vol. 24, no. 4, pp. 1085–1105, 2012.

[9] E. Acar, D. M. Dunlavy, and T. G. Kolda, "A scalable optimization approach for fitting canonical tensor decompositions," *Journal of Chemometrics*, vol. 25, no. 2, pp. 67–86, 2011.

[10] R. A. Harshman, "Foundations of the PARAFAC procedure: Models and conditions for an "explanatory" multimodal factor analysis," *UCLA working papers in phonetics*, vol. 16, 1970.

[11] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," in *Soviet Mathematics Doklady*, 1983, vol. 27, pp. 372–376.

[12] Y. Nesterov, "Gradient methods for minimizing composite functions," *Mathematical Programming*, vol. 140, no. 1, pp. 125–161, Aug 2013.

[13] T. Pock and S. Sabach, "Inertial proximal alternating linearized minimization (iPALM) for nonconvex and nonsmooth problems," *SIAM Journal on Imaging Sciences*, vol. 9, no. 4, pp. 1756–1787, 2016.

[14] Y. Xu, "Alternating proximal gradient method for sparse nonnegative Tucker decomposition," *Mathematical Programming Computation*, vol. 7, no. 1, pp. 39–70, mar 2015, arxiv:1302.2559.

[15] S. Boyd W. Su and E. J. Candès, "A differential equation for modeling nesterov's accelerated gradient method: Theory and insights," *Journal of Machine Learning Research*, vol. 17, no. 153, pp. 1–43, 2016.

[16] M. Muehlebach and M. I. Jordan, "A dynamical systems perspective on nesterov acceleration," *arXiv preprint arXiv:1905.07436*, 2019.

[17] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM Journal on imaging sciences*, vol. 6, no. 3, pp. 1758–1789, 2013.

[18] D. Mitchell, N. Ye, and H. De Sterck, "Nesterov acceleration of alternating least squares for canonical tensor decomposition," 2018.

[19] L. T. K. Hien, N. Gillis, and P. Patrinos, "Inertial block mirror descent method for non-convex non-smooth optimization," 2019, arxiv:1903.01818.

[20] R. Bro, *Multi-way Analysis in the Food Industry: Models, Algorithms, and Applications*, Ph.D. thesis, University of Amsterdam, The Netherlands, 1998.

[21] A. M. S. Ang and N. Gillis, "Accelerating nonnegative matrix factorization algorithms using extrapolation," *Neural computation*, vol. 31, no. 2, pp. 417–439, 2019.

[22] N. Ravindran, N. D. Sidiropoulos, S. Smith, and G. Karypis, "Memory-efficient parallel computation of tensor and matrix products for big tensor decomposition," in *2014 48th Asilomar Conference on Signals, Systems and Computers*. IEEE, 2014, pp. 581–585.

[23] E. E Papalexakis, U. Kang, C. Faloutsos, N. D. Sidiropoulos, and A. Harpale, "Large scale tensor decompositions: Algorithmic developments and applications.," *IEEE Data Eng. Bull.*, vol. 36, no. 3, pp. 59–66, 2013.

[24] A. Ang, J. E. Cohen, and N. Gillis, "Accelerating approximate nonnegative canonical polyadic decomposition using extrapolation," in *XXVIIe Colloque GRETSI*, 2019.