Accelerating Approximate Nonnegative Canonical Polyadic Decomposition using Extrapolation

Andersen ANG[1], Jeremy E. COHEN[2], Nicolas GILLIS[1,*]

[1]Université de Mons, Rue de Houdain 9, 7000, Mons, Belgium

[2]Université de Rennes, INRIA, CNRS, IRISA, France

manshun.ang@umons.ac.be jeremy.cohen@irisa.fr nicolas.gillis@umons.ac.be

Résumé – On considère dans cet article le problème de la Décomposition Positive approchée Canonique Polyadique (aNCPD) d'un tenseur positif d'ordre trois. Ce problème consiste à minimiser $\|\mathcal{T}-(U\otimes V\otimes W)\mathcal{I}_r\|_F^2$ part rapport à des matrices positives U, V et W. Nous proposons un algorithme de descente par blocs de coordonnées utilisant une extrapolation à la Nesterov, de la forme $U^{k+1}=U^k+\beta_k(U^k-U^{k-1})$. Les résultats expérimentaux prouvent l'efficacité de l'algorithme proposé par rapport aux autres approches de descente par blocs pour des données simulées mal conditionnées.

Abstract – In this work, we consider the problem of approximate Nonnegative Canonical Polyadic Decomposition (aNCPD) of third-order nonnegative tensors, which boils down to minimizing $\|\mathcal{T}-(U\otimes V\otimes W)\mathcal{I}_r\|_F^2$ over element-wise nonnegative matrices U, V and W. We present an accelerated block coordinate descent algorithm that uses Nesterov-like extrapolation in the form $U^{k+1}=U^k+\beta_k(U^k-U^{k-1})$. Experimental results showcase the effectiveness of the proposed algorithm with respect to other block-coordinate descent algorithms on ill-conditioned synthetic data.

1 Introduction

Given a nonnegative three dimensional tensor $\mathcal{T}\in\mathbb{R}_+^{I\times J\times K}$, assume one wants to compute the approximation of $\mathcal{T}$ by a nonnegative tensor $\mathcal{G}=(U\otimes V\otimes W)\mathcal{I}_r$ of given small nonnegative rank r, where $\mathcal{G}_{ijk}=\sum_{q=1}^{r}U_{iq}V_{jq}W_{kq}$. Using the Frobenius norm as an error metric, the problem becomes :

$$\min_{U\geq 0,V\geq 0,W\geq 0} f(U,V,W)=\frac{1}{2}\|\mathcal{T}-(U\otimes V\otimes W)\mathcal{I}_r\|_F^2, \quad (1)$$

where the inequalities are meant element-wise.

Problem (1) is often referred to as the approximate Nonnegative Canonical Polyadic Decomposition (aNCPD) of $\mathcal{T}$, as long as r is smaller than the rank of $\mathcal{T}$. This problem is the extension of the low-rank nonnegative approximation problem for matrices, namely *nonnegative matrix factorization* (NMF). Solving aNCPD is an important issue for mining information out of tensors collected in a wide variety of applications; see [7] and references therein. Note that (1) is well-posed. In fact, a best low-rank approximation for tensors may not exist, but because the nonnegativity constraints are lower bounds for the parameters, this prevents degeneracy and the best nonnegative low-rank approximation always exists [9, 13].

*NG acknowledges of the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlanderen (FWO) under EOS Project no O005318F-RG47, and of the European Research Council (ERC starting grant no 679515).

Contribution In this paper, we are interested in algorithms featuring both low-cost iterations and fast convergence speed. Such fast and cheap algorithms are the backbone of modern machine learning, where both the data and the number of parameters to estimate can be very large. To this end, we first study some existing Block-Coordinate Descent (BCD) algorithms, and present an extrapolated BCD algorithm based on Hierarchical Alternating Least Squares (HALS). To be specific, we adopt the numerical scheme of [1], which has found to be able to significantly improve the convergence speed on the problem of NMF. Experimental results showcase that our approach is faster at producing accurate results than state-of-the-art BCD algorithms in all tested scenarios.

2 Existing block-coordinate descent algorithms for aNCPD

Let us first mention some existing algorithms to compute aNCPD. Problem (1) is non-convex and admits no closed-form solution. A baseline method called ANLS solves aNCPD by alternating minimization on blocs $\{U,V,W\}$ as detailled below. However since computing the gradient of f is simple, and projecting on the nonnegative orthant only requires clipping, some algorithms have been proposed based on projected first-order methods [6, 15]. Compression using Tucker models has also been explored [3].

As mentioned in the introduction, in what follows, we study

algorithms with i) a low cost per iteration, and ii) a high convergence speed. First-order methods are typically slow to converge despite having low complexity. However, it is possible to accelerate significantly first-order methods using extrapolation, which is our main contribution described in Section 3 below. Note that other accelerations are of interest but outside the scope of this paper, such as online, mini-batch or randomized algorithms.

Furthermore, we turn towards a particular set of first-order algorithms coined as Block-Coordinate Descent (BCD), where only a subset of the parameters are updated at each iteration.

2.1 Two types of blocks for BCD

To compute approximate unconstrained tensor decomposition models such as the Canonical Polyadic Decomposition or the Tucker decomposition [7], the Alternating Least Squares (ALS) algorithm is considered a baseline algorithm. In what follows, we first describe the well-known Alternating Nonnegative Least Squares algorithm (ANLS), which is an adaptation of ALS for aNCPD.

2.1.1 A block-coordinate framework : ANLS

In ANLS, the cost function f is minimized over each factor matrix U, V and W alternatively until convergence. For factor matrix U, we solve

$$\min_{U \geq 0} \|T_{[1]} - U\,(V \odot W)^T\,\|_F^2 \tag{2}$$

where matrix $T_{[1]}$ is the first mode unfolding of tensor $\mathcal{T}$ as defined in [4] and $\odot$ is the Khatri-Rao product. Note that (2) is simply an equivalent rewriting of (1), with fixed V and W.

It appears now clearly that f is quadratic with respect to each block U, V and W. In other words, problem (2) is a nonnegative least squares problem in a matrix format. Clearly, ANLS is a BCD algorithm where blocks are $\{U, V, W\}$, in this order.

2.1.2 Solving ANLS with BCD : HALS

There are two categories of algorithms to compute NLS. A first category of algorithms produces the exact solution of the NLS in a finite number of steps, such as the historical active set solution by Lawson and Hanson [8]. The number of step is however not known and can be quite large. Also, each iteration in the active set algorithm requires solving a least squares problem, which can be costly. A second category asymptotically converges towards the optimal solution. These algorithms are only interesting if they converge in a shorter time than the exact ones. Among those, Hierarchical Alternating Least Squares is state of the art [5] when dealing with several, alternated NLS problems.

In this paper, we call HALS-NLS the BCD algorithm solving the NLS problem that alternates on the columns of U in (2) while fixing the others. The solution for the ith column u_i of U requires to solve

$$\min_{u_i \geq 0} \|T_{[1]} - U_{\setminus i}(V_{\setminus i} \odot W_{\setminus i})^T - u_i\,(v_i \boxtimes w_i)^T\,\|_F^2 \tag{3}$$

and has the following closed-form solution

$$u_i = \left[\frac{\left(T_{[1]} - U_{\setminus i}(V_{\setminus i} \odot W_{\setminus i})^T\right)(v_i \boxtimes w_i)}{\|v_i\|_2^2 \|w_i\|_2^2}\right]^+ \tag{4}$$

since (3) boils down to the coordinate-wise minimization of second order polynomial functions over a half space. In (3), matrix $M_{\setminus i}$ stands for a matrix M with the i-th column removed, while $\boxtimes$ stands for the Kronecker product. Operator $[x]^+$ clips negative values in x to zero. The NLS problem is solved by alternatively using update rule (4) on the columns of the matrix being updated in the outer ANLS algorithm until convergence. In what follows, we refer to the whole procedure "ANLS with NLS solved by HALS-NLS" as HALS.

Several important remarks can be done at this stage :

— HALS can be seen itself as a BCD algorithm to solve (1), where blocs are

$$\{u_1, u_2, \ldots, u_r, u_1, \ldots, u_r, v_1, v_2, \ldots, w_1, w_2, \ldots\}.$$

— Because updating the columns of U requires the computation of $T_{[1]}\,(V \odot W)$ which is costly, it is computationally interesting to update the columns of U several times before switching to the columns of V.

— The product $(V \odot W)^T(v_i \boxtimes w_i)$ can be computed efficiently as $(V^T v_i * W^T w_i)$ where $*$ is the element-wise product.

3 Extrapolation for ANLS

Let us now discuss the proposed extrapolation scheme for ANLS, which is an extension of the one proposed in [1].

In each step of the regular ANLS, we update a block of parameters using a NLS solver. For instance, when updating U, we compute $U^{k+1} = \text{NLS}(V^k, W^k; U^k)$ where U^k is the initialization for U in an NLS solver such as HALS-NLS.

Extrapolated ANLS (E-ANLS) involves pairing variables $\tilde{U}$, $\tilde{V}$ and $\tilde{W}$ initialized equal to U, V and W. At each iteration, *e.g.*, for U, we compute

$$\text{Update } U^{k+1} = \text{NLS}(\tilde{V}^k, \tilde{W}^k; \tilde{U}^k) \tag{5}$$

$$\text{Extrapolate } \tilde{U}^{k+1} = U^{k+1} + \beta_k(U^{k+1} - U^k), \tag{6}$$

where k is the current iteration index and β_k is the extrapolation parameter. Similar updates are carried out for V and W.

Intuitively, extrapolation makes use of the geometry of the cost function to predict estimates for a future iteration using previous estimates. In a convex optimization setting, Nesterov provided an optimal way to compute β_k for steps (6) in terms of convergence speed. He showed that for this choice of β_k, the error of the iterates of accelerated projected gradient methods converge in $\mathcal{O}(\frac{1}{k^2})$ instead of $\mathcal{O}(\frac{1}{k})$ [11].

However, as aNCPD (1) is non-convex, and because we are using extrapolation with the same β_k for different blocks, there is *a priori* no closed-form solution to update β_k. Thus, the critical part of the update is the parameter β_k, which we now discuss.

3.1 Update of β

The extrapolation scheme [1] has 5 variables, two of them indexed by the iteration index k : $\beta_k, \bar{\beta}_k, \gamma, \bar{\gamma}$ and η, where $\beta_k \in [0,1]$ is the extrapolation variable, $\bar{\beta}_k \in [0,1]$ is the ceiling parameter of β_k. Parameters γ (η) control the way β grows (decays). Lastly $\bar{\gamma}$ is the growth parameter for $\bar{\beta}_k$. We grow or decay β_k after each iteration as follows : let e^k denote the error at iteration k after the extrapolation, then

$$\beta_{k+1} = \begin{cases} \min\{\gamma\beta_k, \bar{\beta}\} & \text{if } e^{k+1} \leq e^k, \\ \beta_k/\eta & \text{else.} \end{cases} \quad (7)$$

It is important to note that extrapolation may increase the objective function value (e^k) due to a bad parameter β_k. If this occurs, we abandon the extrapolation at that iteration and simply set the paring variable $\tilde{U}^{k+1} = U^{k+1}$. In other words, when the error increases, we perform *restart* [12]. In addition, an increase of e^k means the current β_k is too large, we then decrease β_k as (7). To avoid β_{k+1} to grow back to the large value which caused the increase of the error, we update $\bar{\beta}_k$ by (8)

$$\bar{\beta}_{k+1} = \begin{cases} \min\{\bar{\gamma}\bar{\beta}_k, 1\} & \text{if } e^{k+1} \leq e^k \text{ and } \bar{\beta}_k < 1, \\ \beta_k & \text{else.} \end{cases} \quad (8)$$

The relations between the parameters are

$$\forall k \geq 0,\ 0 \leq \beta_k \leq \bar{\beta}_k \leq 1 < \bar{\gamma} \leq \gamma \leq \eta.$$

3.2 An existing line-search approach for ANLS

It is often reported in the literature on computing approximate CPD that ALS is a viable algorithm only when using "line search" [2]. Although it is not stated explicitly in that reference, what is actually done is extrapolation [1]. It is also reported that "line search" speeds up NLS for aNCPD considerably. The scheme is the following heuristic :

$$\text{Update } U^{k+\frac{1}{2}} = NLS(V^k, W^k; U^k) \quad (9)$$

$$\text{Extrapolate } U^{k+1} = U^{k+\frac{1}{2}} + (k^{\frac{1}{h(k)}} - 1)(U^{k+\frac{1}{2}} - U^k) \quad (10)$$

where $h(k)$ is a recursive function so that $h(k+1) = h(k)$ if the error has not increased for more than four iterations, $h(k+1) = 1 + h(k)$ otherwise, and $h(1) = 3$. Also, it is reported that for constrained optimization, "line-search" should be discarded in the first few iterations (4 in tests below) because of instability issues. The main difference between this approach, which we refer to as "Bro" after the name of its author, and the extrapolation scheme we propose is that on top of being based on the Nesterov acceleration, the variables after our scheme are always nonnegative (only the pairing variables may have negative entries).

4 Experiments

Let us now compare various accelerated BCD techniques for aNCPD, namely Projected gradient (PG), HALS (baseline), HALS with extrapolation using Bro's β tuning procedure, E-HALS (proposed), Accelerated Projected Gradient [15] (APG). Notice that the HALS-based algorithms haves inner loops within the HALS and the PG-based algorithms have no inner loop. For fair comparisons we also consider PG and APG with inner loops, namely PG-r and APG-r. We compare these methods in three scenarios described below. In all tests, the rank r is small enough for uniqueness of factor matrices to hold up to trivial permutations and scaling.

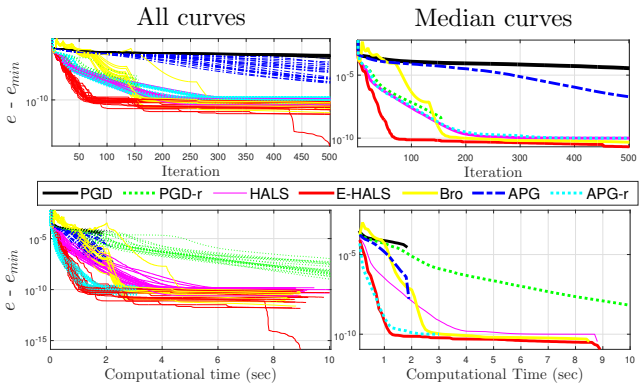

FIGURE 1 – The error curves $e^k - e_{\min}$ of various algorithms, where $e_{\min}$ is the lowest e among all trials of all algorithms.

Set up We form the ground truth tensor T^{true} by sampling elements of matrices U^{true},V^{true} and W^{true} from the uniform distribution $\mathcal{U}[0,1]$. To make the problem more difficult, we increase the condition number of the mode-1 matrix U by replacing $U^{\text{true}}(:,1)$ with $0.01U^{\text{true}}(:,1) + 0.99U^{\text{true}}(:,2)$ (using Matlab array notations) and we add i.i.d. Gaussian noise to the tensor with variance $\sigma^2 = 10^{-4}$. Note that some values in the noisy tensor may be negative due to the noise. We run all the algorithms under the same number of iteration (500) with maximum run time of 10 seconds, same number of inner loops (50) and random initialization U^{ini},V^{ini},W^{ini} sampled from $\mathcal{U}[0,1]$. For E-HALS, we set $\bar{\beta}_0 = 1$, $\beta_0 = 0.4$, $\gamma = 1.1$, $\bar{\gamma} = 1.001$ and $\eta = 2$ after calibration. We repeat the whole process 20 times. We perform the following three tests :

1. Low rank, balanced sizes We set $I = J = K = 50, R = 10$. Fig. 1 shows the cost function values for each algorithm against iteration and computational time. Table 1 provides the median reconstruction error of each mode compared to ground truth : we define RE_{final} as the relative error between U^k and U^{true} in percent after column normalization and matching. The unpredictable behavior of Bro (error may increase significantly) shows that it is not *a priori* well-designed for aNCPD, even though its final error is sometimes on par with other algorithms. APG-r performs better than HALS, while E-HALS is faster than the other methods.

2. Low rank, balanced sizes, ill-conditioned We use the same setting as in the previous test. However here the condition number of factor matrix U is severely increased using $U = U(I_r + \mathbf{1}_r)$, where I_r is identity matrix of size r and $\mathbf{1}_r$ is r-by-r all-1 matrix. Due to U being very ill-conditioned, method Bro di-

1. Line-search as usually understood in smooth optimization has been studied for approximate CPD in [14]

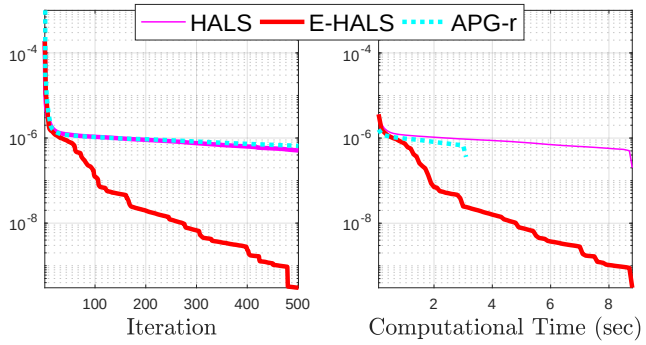

FIGURE 2 – Median value of $e^k - e_{\min}$ of HALS, E-HALS and APG-r for test 2.

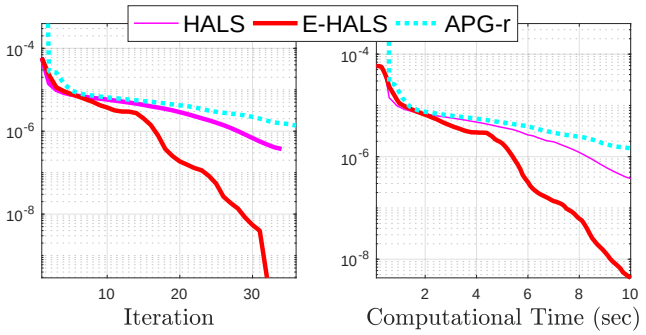

FIGURE 3 – Median value of $e^k - e_{\min}$ of HALS, E-HALS and APG-r for test 3.

verges (it therefore does not show on Fig. 2). On the other hand, E-HALS is not only the fastest method, see Fig. 2, but also provides much more accurate factor estimates, see Table 1. Unexpectedly, E-HALS seems to perform even better in this second test than in the previous one.

3. Medium rank, unbalanced sizes Here $I = 150$, $J = 10^3$, $K = 35$ and $R = 20$. Fig. 3 shows similar results to the previous tests : E-HALS outperforms all other BCD approaches.

TABLE 1 – Median RE_{final} in % of U, V, W, (* means ≥ 40)

Algo	Test 1	Test 2	Test 3
PGD	15, 1.8, 1.8	4.3, *, *	*, *, *
PGD-r	0.2, 1.8, 1.6	4.1, *, *	*, *, *
HALS	0.2, 1.6, 1.6	2.2, 22, 23	4.7, 5.2, 5.2
E-HALS	0.2, 1.8, 1.8	**0.04,0.3, 0.3**	**0.4,0.8,0.8**
Bro	0.2, 1.5, 1.5	0.2, 2.4, 2.4	*, *, *
APG	0.5, 3.0, 2.9	4.3, *, *	*, *, *
APG-r	**0.2, 1.3, 1.2**	2.7, *, 28	10,11,11

5 Conclusion

In this paper, we presented an algorithm to compute approximate Nonnegative Canonical Polyadic Decomposition that employs extrapolation to accelerate the convergence of a block-coordinate scheme. The extrapolation is in the form of $U^{k+1} = U^k + \beta_k(U^k - U^{k-1})$ where k is the iteration number and $\beta_k \geq 0$ is the extrapolation parameter. Experiment results show that the proposed scheme can significantly speed up the algorithm and is much faster than benchmark algorithms in several ill-conditionned settings. Future works will further improve on speed and adapt the proposed method to other decompositions such as approximate nonnegative Tucker. Also, we will adapt other acceleration methods desgigned in the unconstrained case (here we only compare with Bro [2]), such as the one in [10].

Références

[1] A. M. S. Ang and N. Gillis. Accelerating nonnegative matrix factorization algorithms using extrapolation. *Neural computation*, 31(2) :417–439, 2019.

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

[3] J. E. Cohen, R. Cabral-Farias, and P. Comon. Fast decomposition of large nonnegative tensors. *IEEE Signal Processing Letters*, 22(7) :862–866, July 2015.

[4] Jeremy E Cohen. About notations in multiway array processing. *arXiv preprint arXiv :1511.01306*, 2015.

[5] N. Gillis and F. Glineur. Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. *Neural computation*, 24(4) :1085–1105, 2012.

[6] K. Huang, N. D. Sidiropoulos, and A. P. Liavas. A flexible and efficient algorithmic framework for constrained matrix and tensor factorization. *IEEE Transactions on Signal Processing*, 64(19) :5052–5065, 2016.

[7] T. G. Kolda and B. W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3) :455–500, September 2009.

[8] C. L. Lawson and R. J. Hanson. *Solving least squares problems*, volume 15. Siam, 1995.

[9] L-H. Lim and P. Comon. Nonnegative approximations of nonnegative tensors. *J. Chemom.*, 23(7-8) :432–441, 2009.

[10] Drew Mitchell, Nan Ye, and Hans De Sterck. Nesterov acceleration of alternating least squares for canonical tensor decomposition. *arXiv preprint arXiv :1810.05846*, 2018.

[11] Y. Nesterov. *Introductory lectures on convex optimization : A basic course*. Springer Science & Business Media, 2013.

[12] B. O'donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Foundations of computational mathematics*, 15(3) :715–732, 2015.

[13] Y. Qi, P. Comon, and L. H. Lim. Uniqueness of nonnegative tensor approximations. *IEEE Trans. Inf. Theory*, 62(4) :2170–2183, April 2016. arXiv :1410.8129.

[14] M. Rajih, P. Comon, and R. A. Harshman. Enhanced line search : A novel method to accelerate parafac. *SIAM journal on matrix analysis and applications*, 30(3) :1128–1147, 2008.

[15] 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*, 6(3) :1758–1789, 2013.