Accelerating Block-coordinate Descent Algorithms for (Nonnegative) Tensor Factorization

Man Shun A. Ang, **Jeremy E. Cohen**, Le Thi Khanh Hien and Nicolas Gillis

Univ. Waterloo, **IRISA, CNRS**, UMONS

21 Mai 2021

Roadmap

1. An introduction to approximate Nonnegative CP Decomposition

2. Reminders on nonnegative least-squares and extrapolation

3. Contribution: HER-BCD algorithm

A team effort

Andersen Ang	Thi Khanh Hien Le	Nicolas Gillis
Post-doc, Univ. Waterloo	Post-doc, UMONS	Ass. Prof, UMONS

A. M. S. Ang, J. E. Cohen, N. Gillis, L. T. K. Hien, "Accelerating Block Coordinate Descent for Nonnegative Tensor Factorization", Numerical Linear Algebra Appl., 2021;e2373.

Tensor decomposition models are sums of rank-one terms

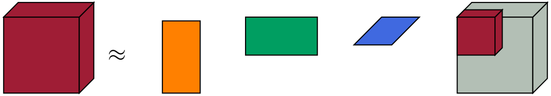
Canonycal Polyadic Decomposition:

$$\mathcal{T} = \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q$$

$$\mathcal{T} \quad = \quad a_1 \otimes b_1 \otimes c_1 + \cdots + a_r \otimes b_r \otimes c_r$$

Tucker Decomposition:

$$\mathcal{T} = \sum_{q_1,q_2,q_3=1}^{r_1,r_2,r_3} g_{q_1 q_2 q_3} a_{q_1} \otimes b_{q_2} \otimes c_{q_3}$$



$$\mathcal{T} \quad \approx \quad (A \otimes B \otimes C) \quad \mathcal{G}$$

Definition: tensor [CP] rank

$\text{rank}(\mathcal{T}) = \min\{r \mid \mathcal{T} = \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q\}$

Tensor CP rank coincides with matrix “usual” rank!

Making use of low-rank representations

Let $A = [a_1, a_2, \dots, a_r]$, B and C similarly built.

Uniqueness of the CPD

Under mild conditions

$$krank(A) + krank(B) + krank(C) - 2 \geq 2r,$$

the CPD of $\mathcal{T}$ is essentially unique (i.e.) the rank-one terms are unique.

This means we can interpret the rank-one terms a_q, b_q, c_q
$\rightarrow$ Source Separation!

Compression (also true for other models)

The CPD involves $r(I + J + K - 2)$ parameters, while $\mathcal{T}$ contains IJK entries.

If the rank is small, this means huge compression/dimensionality reduction for function approximation.

- missing values completion, denoising
- imposing sparse structure to solve other problems (PDE, neural networks, dictionary learning...)

Approximate CPD

- Often, $\mathcal{T} \approx \sum_q^r a_q \otimes b_q \otimes c_q$ for small r.
- However, the generic rank (i.e. rank of random tensor) is very large.
- Therefore if $\mathcal{T} = \sum_q^r a_q \otimes b_q \otimes c_q + \mathcal{N}$ with $\mathcal{N}$ some small Gaussian noise, it has approximatively rank lower than r but its exact rank is large.

Best low-rank approximate CPD

For a given rank r, the cost function

$$\eta(A, B, C) = \|\mathcal{T} - \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q\|_F^2$$

has the following properties:

- it is infinitely differentiable.
- it is non-convex in (A, B, C), but quadratic in A or B or C.
- its minimum may not be attained (ill-posed problem).

Approximate Nonnegative CPD

Low-rank r approximate NCPD

Given a tensor $\mathcal{T}$, find tensor $\mathcal{G}^* = \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q$ that minimizes

$$\eta(A, B, C) = \|\mathcal{T} - \sum_{q=1}^{r} a_q \otimes b_q \otimes c_q\|_F^2 \text{ so that } a_q \geq 0, b_q \geq 0, c_q \geq 0$$

- The minimum is always attained (coercivity)!
- The cost is not smooth anymore.

Well-posedness

Approximate NCPD is well posed:

- the best low nonnegative rank approximation $\mathcal{G}^*$ exists. [Lim, Comon 2009]
- *most of the time*, tensor $\mathcal{G}^*$ is unique [Qi, Lim, Comon 2016]

My favorite class of algorithms to solve aNCPD: block-coordinate descent!

Nonconvex optimization algorithms, an incomplete list

All at once

- Conjugate gradient
- ADMM
- Levenberg Marquardt and others

nonnegativity imposed by interior point methods, squaring or active set.

X ADMM $<$ AOADMM, PG $<$ APG

X Typically slower than BCD

O Very efficient near optimum

Block coordinate (alternating)

- Alternating proximal gradient
- Alternating nonnegative least squares (ANLS)
- HALS
- Multiplicative updates
- AOADMM

nonnegativity imposed mostly by proximal step.

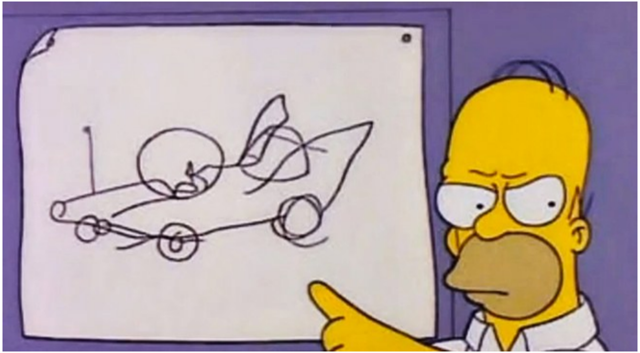
O Easy to design and implement

O Convex optimization tools

O Fast in practice

Problematic

Be cheap, be fast.



How to make tensor algorithms faster?

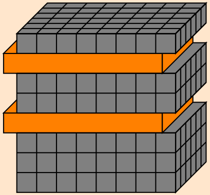
HPC

Not my expertise...

- n-mode product
- NNLS
- ??

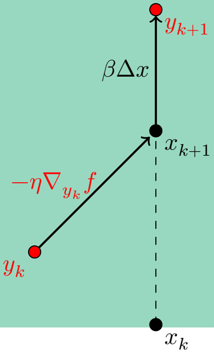
Sampling and Randomization

- Compression
- Sketching
- Subtensor sampling
- Fiber sampling
- Element-wise sampling



Acceleration

- Adagrad
- Momentum
- Quantification
- Extrapolation



Roadmap

1. An introduction to approximate Nonnegative CP Decomposition
2. Reminders on nonnegative least-squares and extrapolation
3. Contribution: HER-BCD algorithm

Reminder 1: Alternating nonnegative least squares for aNCPD

Problem:

$$\underset{a_q\geq 0, b_q\geq 0, c_q\geq 0}{argmin} \|\mathcal{T} - \textstyle\sum_{q=1}^{r} a_q \otimes b_q \otimes c_q\|_F^2$$

Equivalent problem:

$$\underset{A\geq 0, B\geq 0, C\geq 0}{argmin} \|T_{[1]} - A(B \odot C)^T\|_F^2 \underset{\text{fix} B,C}{\rightarrow} \underset{A\geq 0}{argmin} \|T_{[1]} - A(B \odot C)^T\|_F^2$$

where $T_{[1]}$ is an unfolding of $\mathcal{T}$ and $\odot$ is the Khatri Rao product and $A = [a_1, \dots, a_r]$.

The ANLS algorithm (or any typical BCD algorithm)

loop until convergence:

- Update A using NNLS($T_{[1]}, B \odot C$)
- Update B using NNLS($T_{[2]}, A \odot B$)
- Update C using NNLS($T_{[3]}, A \odot C$)

Reminder 2: NonNegative Least Squares

U update problem: NNLS

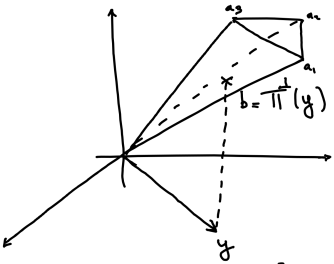
$$\underset{X\geq 0}{argmin}\ \|Y-AX\|_F^2$$

Convex!

Algorithms:

- Active set [Lawson Hanson 1974, Bro 1997]
- **Hierarchical Alternating Least Squares (HALS)**
- Block Principal Pivoting [Kim Park 2011]
- Any proximal gradient method

Note: HALS is also a BCD algorithm.



$$b = \underset{z \in cd_+(A)}{argmin}\ \|y-z\|_2^2 = Ax$$

Reminder 3: Nesterov extrapolation for convex optimization

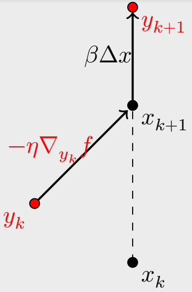
Given a (strongly) convex differentiable form f, L Lipschitz continuous, solve

$$\underset{x \in [0,1]^n}{argmin} \; f(x)$$

Fast gradient algorithm (simplified)

- $\eta = 1/L$; initialize x; $y = x$
- loop until convergence:
  1. $x_{old} = x$
  2. $\beta =$ some formula(β)
  3. $x = y - \eta \nabla_y f$
  4. $y = x + \beta(x - x_{old})$

Note: Step 3. can be replaced by a proximal gradient step to account for constraints.

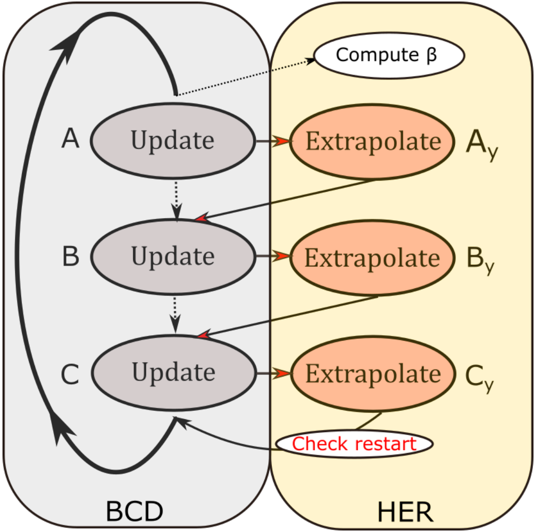


Improves gradient descent convergence rate for strongly convex maps from $\mathcal{O}(\frac{1}{k})$ to $\mathcal{O}(\frac{1}{k^2})$.

Roadmap

1. An introduction to approximate Nonnegative CP Decomposition
2. Reminders on nonnegative least-squares and extrapolation
3. Contribution: HER-BCD algorithm

Contribution: Heuristic Extrapolation in BCD algorithms



Heuristic Extrapolation with Restart (HER)

- Introduce pairing variables
- Update a block, then extrapolate heuristically
- Perform restart if error increases

Different from

- using extrapolation in the updates
- using extrapolation after each outer loop

Extrapolation for ANLS using HALS with restart: E-HALS

The E-HALS algorithm

- initialize A, B, C; $A_y = A, B_y = B, C_y = C$
- loop until convergence:
  1. $A_{old} = A, B_{old} = B; C_{old} = C$
  2. Update β with heuristic (next slide)
  3. Update A using NNLS$(T_{[1]}, B_y \odot C_y)$
  4. Extrapolate $A_y = [A + \beta(A - A_{old})]_+$
  5. Update B using NNLS$(T_{[2]}, A_y \odot C_y)$
  6. Extrapolate $B_y = [B + \beta(B - B_{old})]_+$
  7. Update C using NNLS$(T_{[3]}, A_y \odot B_y)$
  8. Extrapolate $C_y = [C + \beta(C - C_{old})]_+$
- if cost function increases, restart $A_y = A, B_y = B, C_y = C$

A remark on restart

At each iteration,

1. if error has decreased, increase β up to a threshold β_{max}.
2. if error has increased, restart, decrease β and β_{max}.

In any case, $\beta \in]0, \beta_{max}]$ with $\beta_{max} \leq 1$.

To perform restart, denoting $F(A, B, C) = \|T_{[3]} - C(A \odot B)^T\|_F^2$, we check if

$$F(A_y^k, B_y^k, C^k) < F(A_y^{k-1}, B_y^{k-1}, C^{k-1}).$$

Using pairing variables Ay, By instead of A, B allows to save computation time.

Experimental Results: setup

Balanced dimensions

- $r = 10$
- $I = J = K = 50$
- Uniform A, B, C
- noiseless

Unbalanced dimensions

- $r = 12$
- $I = 150$
- $J = 10^3$
- $K = 35$
- Uniform A, B, C
- noiseless

Difficulty:

We test with HALS and ADMM nnls solvers, more in the paper!

Plots

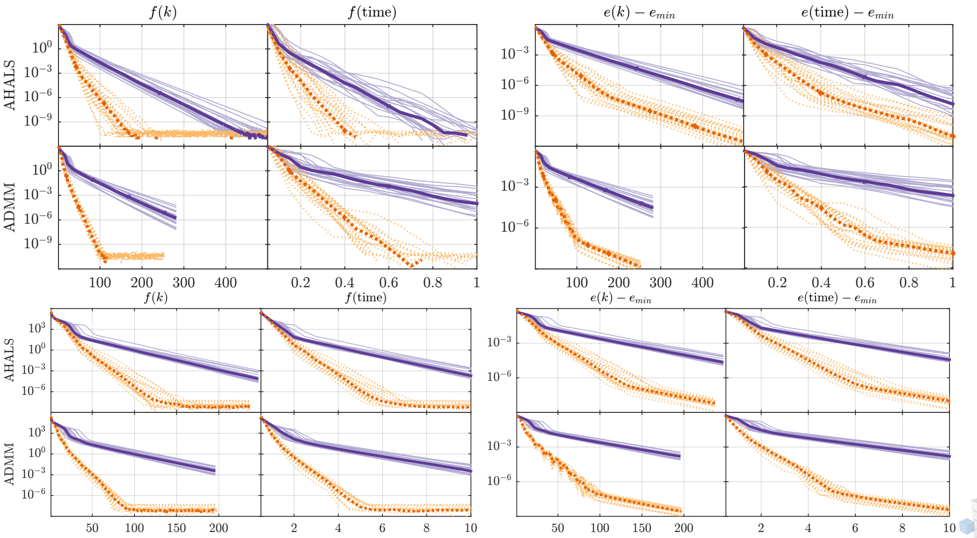


Figure: Convergence of algorithms : A-HALS and AO-ADMM without HER (solid purple) and with HER (dotted orange). Balanced dimensions, ill-conditionned factors

A few other extrapolation methods

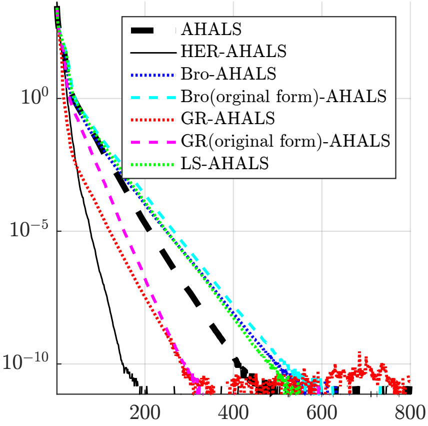


Figure: Comparing AHALS with different acceleration frameworks on synthetic datasets

Conclusion and perspectives

Conclusions

- A heuristic to extrapolate BCD algorithms for tensor decomposition is proposed.
- It accelerates all BCD algorithms we could try.
- Sometimes no sensible acceleration.
- Costs virtually nothing.

Perspectives

- Integration within sketching methods? [Tried CPRAND, mitigated results]
- Convergence proof? Under which assumptions?
- Try on other problems solved with BCD?

Thank you for your attention