2019 IEEE INTERNATIONAL WORKSHOP ON MACHINE LEARNING FOR SIGNAL PROCESSING, OCT. 13–16, 2019, PITTSBURGH, PA, USA

ACCELERATING NONNEGATIVE MATRIX FACTORIZATION OVER POLYNOMIAL SIGNALS WITH FASTER PROJECTIONS

*Cécile Hautecoeur, François Glineur**

Université catholique de Louvain
CORE and ICTEAM Institute
B-1348 Louvain-la-Neuve - Belgium
cecile.hautecoeur@uclouvain.be - francois.glineur@uclouvain.be

ABSTRACT

Nonnegative Matrix Factorization is a data analysis tool that aims at representing a set of input data vectors as nonnegative linear combinations of a few nonnegative basis vectors. When dealing with continuous input signals, smoothness and accuracy of this representation can often be improved if nonnegative polynomials are used in the basis instead of vectors. However, algorithms using polynomials are usually more computationally demanding than their vector counterparts.

In this work, we consider the Hierarchical Alternating Least Squares method, which displays state-of-the art performance on this problem and requires at each iteration to compute projections over the set of nonnegative polynomials. We introduce several heuristic algorithms designed to provide fast approximations of these projections, and show that their use significantly accelerates the resolution of nonnegative matrix factorization problems over polynomial signals without adverse effect on the accuracy of the obtained solutions.

Index Terms— Nonnegative matrix factorization, polynomial signals, nonnegative polynomials, hierarchical alternating least squares, projection over nonnegative polynomials

1. INTRODUCTION

Nonnegative Matrix Factorization (NMF) is a central tool to deal with nonnegative data in order to perform data compression, data analysis or data completion. This framework is able to extract characteristic and meaningful features from a data set while filtering noise [1], and is used for various kinds of data such as images [2, 3] or chemical and physical spectra [4, 5]. Traditionally, NMF represents input data as linear combinations of characteristic vectors: given a data matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$ and a rank $r \ll \min\{m,n\}$, one seeks two nonnegative matrices $\boldsymbol{A} \in \mathbb{R}^{m\times r}$ and $\boldsymbol{X} \in \mathbb{R}^{r\times n}$ such that $\boldsymbol{Y} \simeq \boldsymbol{AX}$. Using the Frobenius norm to measure the accuracy of the approximation, this problem can be written

$$\min_{\boldsymbol{A}\in\mathbb{R}^{m\times r}_+,\ \boldsymbol{X}\in\mathbb{R}^{r\times n}_+} ||\boldsymbol{Y}-\boldsymbol{AX}||_F^2 \,. \qquad \text{(NMF)}$$

In this work, we consider NMF of polynomial signals (P-NMF), a similar problem where the r basis elements contained in the columns of $\boldsymbol{A}$ are (discretized) polynomials. Indeed, as NMF is in general non unique [6], it can be improved using a priori knowledge on the data [7, 8]. Since many continuous signals can be well approximated using polynomials, approximate signals as combinations of polynomials provide smooth continuous approximations (valid not only at some discretization points).

Such an approach, using polynomials or splines, has been considered in the literature [9, 10, 11, 12, 13] and, besides the advantages presented above, can offer a more efficient way to deal with (very) large-scale data compared to usual approaches (using low-degree polynomials or a few splines to represent signals discretized over many points). One of the best performing methods for P-NMF, recently proposed in [13], is a generalization of the Hierarchical Alternating Least Squares (HALS) method [14] that displays state-of-the-art performance for standard NMF. However, it relies on the repeated computation of projections over the set of nonnegative polynomials at each iteration, which can dominate its computational effort (as shown in Section 2). Therefore, we introduce in Section 3 several fast heuristic algorithms designed to approximate this projection and show that they very significantly accelerate the HALS algorithm for P-NMF, while attaining essentially the same final accuracy, as observed in numerical experiments reported in Section 4.

2. NMF OVER POLYNOMIAL SIGNALS

Suppose that the data to approximate are a set of n continuous univariate functions computed on m discretization points $\{\tau_j\}_{j=1}^m$ belonging to a fixed interval $[a,b]$, and gathered in input matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$. Given a rank r, and a degree d,

*This work was supported by the Fonds de la Recherche Scientifique - FNRS and the Fonds Wetenschappelijk Onderzoek - Vlaanderen under EOS Project no 30468160.

978-1-7281-0824-7/19/$31.00

our goal is to find r polynomials of degree d, nonnegative over interval $[a, b]$, such that every input function in $\boldsymbol{Y}$ can be approximated by a linear combination of those polynomials. More precisely, one seeks the values of those polynomials at the discretization points $\{\tau_j\}$, gathered as columns in matrix $\boldsymbol{A} \in \mathbb{R}^{m\times r}$, and a nonnegative mixing matrix $\boldsymbol{X} \in \mathbb{R}_+^{r\times n}$ such that $\boldsymbol{Y} \simeq \boldsymbol{AX}$.

Every polynomial of degree d can be expressed as a linear combination of $d+1$ basis polynomials $\{\pi_0, \cdots, \pi_d\}$ (we use a Chebyshev basis in this work). Then, the discretized values of a polynomial $f(t)$ defined by a vector of coefficients $\boldsymbol{f}$ can be expressed as $\boldsymbol{\Pi f}$, where $\boldsymbol{\Pi} \in \mathbb{R}^{m\times(d+1)}$ is a Vandermonde-like matrix (that depends on points $\{\tau_j\}$). Hence matrix $\boldsymbol{A}$ containing discretized values of polynomials can be expressed as $\boldsymbol{\Pi B}$, where columns of matrix $\boldsymbol{B}$ (denoted by $\boldsymbol{b_{:i}}$) contain the coefficients of those polynomials in the chosen basis. Let $\mathcal{P}_+^d(I)$ be the set of coefficient of degree d polynomials nonnegative over interval I, our problem can be formulated as

$$\min_{\boldsymbol{B},\boldsymbol{X}} \ ||\boldsymbol{Y} - \boldsymbol{\Pi B X}||_F^2 \text{ such that}$$
$$\boldsymbol{b_{:j}} \in \mathcal{P}_+^d([a,b]),\ x_{ji} \geq 0 \quad \forall 1 \leq j \leq r, 1 \leq i \leq n.$$

which can be rewritten as follows, with the help of two matrices $\boldsymbol{Z} = \boldsymbol{\Pi}^\top \boldsymbol{Y} \in \mathbb{R}^{(d+1)\times n}$ and $\boldsymbol{M} = \boldsymbol{\Pi}^\top \boldsymbol{\Pi} \in \mathbb{R}^{(d+1)\times(d+1)}$ (both can be precomputed)

$$\min_{\boldsymbol{B},\boldsymbol{X}} \sum_i -2\boldsymbol{z_{:i}}^\top \boldsymbol{B x_{:i}} + \boldsymbol{x_{:i}}^\top \boldsymbol{B}^\top \boldsymbol{M B x_{:i}} \qquad \text{(P-NMF)}$$
$$\text{such that } \boldsymbol{b_{:j}} \in \mathcal{P}_+^d([a,b]),\ x_{ji} \geq 0 \quad \forall j, i.$$

2.1. Methods to solve the P-NMF problem

Least squares approach (P-LS). The P-NMF problem is solved in [11] using a nonlinear least-squares solver. To do so, nonnegativity constraints are enforced using well-chosen parametrizations: nonnegative entries in mixing matrix $\boldsymbol{X}$ are represented with $x_{i,j} = h_{i,j}^2$, while nonnegativity of polynomials in $\boldsymbol{B}$ is obtained from the following characterization [15] of even-degree polynomial nonnegative over $[-1, 1]$

$$g(t) \geq 0 \text{ for } t \in [-1,1] \Leftrightarrow g(t) = a(t)^2 + (1-t^2)b(t)^2 \quad (1)$$

where $a(t)$ has degree $d/2$, and $b(t)$ has degree $d/2-1$ (Markov-Lukacs). Note that a similar representation holds for polynomials of odd degree, and that it can be extended to any interval $[a, b]$ using an appropriate change of variables.

Hierachical Alternating Least Square approach (P-HALS). The P-NMF problem was also solved in [13] using the Hierarchical Alternating Least Squares method (HALS) [14], which consists in alternatively updating the rows of $\boldsymbol{X}$ and the columns of $\boldsymbol{A}$ using their exact minimizers for the objective function. To maintain nonnegativity, negative values in the $\boldsymbol{X}$ minimizers are replaced by zero, which can be seen as a projection over the set of nonnegative vectors ($[\xi]_+ = \max\{\xi, 0\}$). Similarly, one can enforce nonnegativity of the polynomials contained in the columns of $\boldsymbol{A}$ by projecting them over the set of polynomials nonnegative over interval $[a, b]$, an operation we denote $[\cdot]_{\mathcal{P}_+^d([a,b])}$. This leads to the following updates

$$\boldsymbol{b_{:k}} \to \left[\frac{(\boldsymbol{M})^{-1}\boldsymbol{Z x_{k:}}^\top - \sum_{j\neq k} \boldsymbol{b_{:j} x_{j:} x_{k:}}^\top}{\boldsymbol{x_{k:} x_{k:}}^\top}\right]_{\mathcal{P}_+^d([a,b])}$$

$$\boldsymbol{x_{k:}} \to \left[\frac{\boldsymbol{b_{:k}}^\top \boldsymbol{Z} - \sum_{j\neq k} \boldsymbol{b_{:k}}^\top \boldsymbol{M b_{:j} x_{j:}}}{\boldsymbol{b_{:k}}^\top \boldsymbol{M b_{:k}}}\right]_+ \qquad \text{(P-HALS)}$$

(noting that polynomials are updated through coefficient matrix $\boldsymbol{B}$). This method is presented in Algorithm 1.

Algorithm 1: P-HALS updates

Function `updateB(`$\boldsymbol{M_1} = \boldsymbol{M}^{-1}\boldsymbol{Z}$, $\boldsymbol{X}$, $\boldsymbol{M}$`)`**:**
- $\boldsymbol{P} = \boldsymbol{M_1 X}^\top$, $\boldsymbol{Q} = \boldsymbol{X X}^\top$
- **foreach** $\boldsymbol{b_{:j}} \in \boldsymbol{B}$ **do**
  - $C = \boldsymbol{p_{:j}} - \sum_{k\neq j} \boldsymbol{b_{:k}} q_{kj}$
  - $\boldsymbol{b_{:j}} \leftarrow$ `NonnegativePol`(C/q_{jj})
- **return** $\boldsymbol{B}$

Function `updateX(`$\boldsymbol{Z}$, $\boldsymbol{M}$, $\boldsymbol{B}$`)`**:**
- $\boldsymbol{P} = \boldsymbol{B}^\top \boldsymbol{Z}$, $\boldsymbol{Q} = \boldsymbol{B}^\top \boldsymbol{M B}$
- **foreach** $\boldsymbol{x_{j:}}$ ***in*** $\boldsymbol{X}$ **do**
  - $C = \boldsymbol{p_{j:}} - \sum_{k\neq j} q_{jk} \boldsymbol{x_{k:}}$
  - $\boldsymbol{x_{j:}} \leftarrow \max(0, C/q_{jj})$
- **return** $\boldsymbol{X}$

2.2. Projection over the set of nonnegative polynomials

To project a polynomial $f(t)$ over the set of polynomials nonnegative on $[a, b]$, we need to compute $g(t) \in \mathcal{P}_+^d([a, b])$ that is as close as possible to $f(t)$ over the discretization points $\{\tau_j\}_{j=1}^m \in [a, b]$. Using Cholesky factorization on positive semi-definite matrix $\boldsymbol{M}$, introduced for P-NMF, as $\boldsymbol{M} = \boldsymbol{L}^\top \boldsymbol{L}$, and representing both $f(t)$ and $g(t)$ with their coefficient vectors $\boldsymbol{f}$ and $\boldsymbol{g}$, the projection can be written as:

$$\min_{\boldsymbol{g}} ||\boldsymbol{L}(\boldsymbol{f} - \boldsymbol{g})||_2^2 \quad \text{such that } \boldsymbol{g} \in \mathcal{P}_+^d([a,b]). \qquad (2)$$

Even though set $\mathcal{P}_+^d([a, b])$ forms a convex cone, it is impossible to handle directly the constraint in problem (2), as it is infinite-dimensional ($g(t) \geq 0\ \forall t \in [a, b]$). Nevertheless, nonnegative univariate polynomials can be represented using positive semi-definite matrices via the sum-of-squares (SOS) approach [16]. A polynomial $f(t)$ is SOS if and only if

$$f(t) = \textstyle\sum_i g_i(t)^2 \equiv \boldsymbol{\Pi} \sum_i \boldsymbol{g_i g_i}^\top \boldsymbol{\Pi}^\top = \boldsymbol{\Pi Q \Pi}^\top. \quad (3)$$

where $\boldsymbol{Q}$ is a positive semi-definite matrix in $\mathbb{R}^{d/2+1\times d/2+1}$, called the Gram matrix associated to $f(t)$ (and coefficients

$\boldsymbol{f}$ depend linearly from entries of matrix $\boldsymbol{Q}$). Moreover, using a result similar to (1), we can represent any even degree polynomial $g(x) \in \mathcal{P}_+^d([a,b])$ using

$$g(t) \geq 0 \text{ for } t \in [-1,1] \;\Leftrightarrow g(t) = a(t) + (1-t^2)b(t) \quad (4)$$

where $a(t), b(t)$ are both SOS, $a(t)$ has degree d and $b(t)$ has degree $d-2$ (Markov-Lukacs). This can also be extended to odd degree polynomials, and to any interval $[a,b]$. Our projection can then be obtained as the solution of the following tractable (convex, conic) semidefinite optimization problem

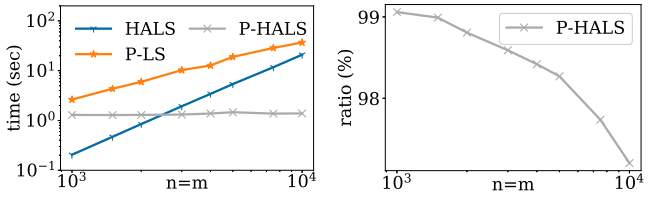
$$\begin{array}{ll} \min t & \text{(PROJECTION)} \\ \text{such that } (\boldsymbol{u}, t) \in \mathbb{L}^{d+1} & \mathbb{L} = \text{Lorentz cone: } ||\boldsymbol{u}|| \leq t \\ \boldsymbol{A} \in \mathbb{S}^{d/2+1}, \boldsymbol{B} \in \mathbb{S}^{d/2} & \mathbb{S} = \text{Semidefinite cone} \\ \boldsymbol{u} = \boldsymbol{L}\Big(\boldsymbol{f} - \ell(\boldsymbol{A}) - \ell'(\boldsymbol{B})\Big) & \boldsymbol{u} = \boldsymbol{L}(\boldsymbol{f} - \boldsymbol{g}) \end{array}$$

(with suitable linear operators ℓ and ℓ').

2.3. Comparison of computational effort to solve P-NMF

Figure 1(a) displays time needed to compute 20 iterations of both P-LS and P-HALS as well as usual HALS (which does not enforce the polynomial structure), for $r = 3$ polynomials of degree $d = 12$, and for a growing number n of observations and number m of discretization points (with $n = m$).

Standard HALS is faster than polynomial-based algorithms for small-scale problems, but P-HALS and later P-LS become competitive as the size of the problem increases. Moreover, iteration times for P-HALS do not vary much with the size of the problem, as the semidefinite optimization-based projections over nonnegative polynomials are by far the most time-consuming part of the algorithm, as seen on Figure 1(b) (more than 97% of the computational time spent). We therefore explore ways to accelerate those polynomial projections over $\mathcal{P}_+^d([a,b])$ in the next section.



(a) Needed time (in seconds) (b) Ratio projection/total time

Fig. 1: Performances of the P-HALS algorithm over large datasets, using $r = 3$, $d = 12$, $10^3 \leq m = n \leq 10^4$ and a signal to noise ratio of 20dB (see Section 4 for details about the used datasets).

3. HEURISTIC ALGORITHMS FOR PROJECTION

We introduce below several new heuristic algorithms designed to replace the costly projection over nonnegative polynomials. They allow to compute a polynomial $\boldsymbol{g}$ that is close to $\boldsymbol{f}$ while constrained to be nonnegative on the set of discretization points $\{\tau_i\}_{i=1}^m$, but they no longer strictly enforce nonnegativity over the whole interval $[a,b]$.

3.1. General heuristics

Discretization heuristic. A first approach consists in solving the projection from (2) with constraint $g(t) \geq 0 \; \forall x \in [a,b]$ relaxed to $g(t) \geq 0 \; \forall t \in \mathcal{D}$ where set $\mathcal{D} \subset [a,b]$ contains a finite number D of points ($|\mathcal{D}| = D$), among which the given discretization points $\{\tau_j\}_{j=1}^m$. This relaxed projection problem becomes a second-order cone optimization problem, and is solved faster than the exact version.

Figure 2 shows how the chosen number D of discretization points impacts this method: both the computation times and the accuracy (norm of the error w.r.t. exact projection) are reported, as well as an estimate of the total proportion of $[a,b]$ where the projection is negative (the lower, the better). Choosing more discretization points significantly slows down the algorithm, but also significantly decreases the projection error as well as the length of negative subintervals.

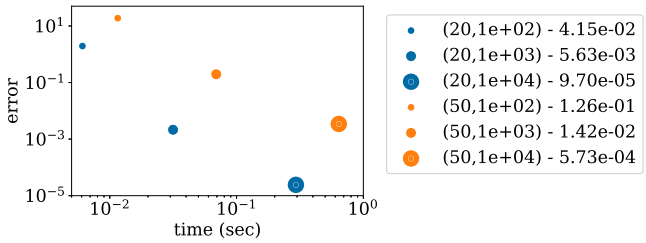


Fig. 2: Influence of discretization parameter D on projection. Bigger symbols for bigger D. Legend indicates (d, D) and an estimate of the length of subintervals where projection is negative.

Two iterative heuristics Our next idea is derived from the fact that polynomial curve fitting is much faster than the projections (exact or approximate) presented in the previous sections, as it corresponds to solving a linear system. Consequently, in order to project a given polynomial $f(t)$, we propose to apply polynomial curve fitting to a list of points obtained from the graph of f with its negative part truncated. Again, we consider a set $\mathcal{D} \subset [a,b]$ containing a finite number of points D among which are the given discretization points ($\{\tau_j\}_{j=1}^m \in \mathcal{D}$). More concretely, we fit a degree-d polynomial to points (λ_i, y_i) ($\forall \lambda_i \in D$) where $y_i = f(\lambda_i)$ for every abscissa λ_i such that $f(\lambda_i) > 0$, and $y_i = \epsilon$ otherwise (with ϵ a small positive number). As the fitted polynomial still contains negative values in general, we propose two iterative approaches to ultimately obtain a nonnegative polynomial:

- H1: At each iteration, use the obtained approximation as initial polynomial for the next fit.
- H2: At each iteration, add ϵ to the values y_i where the obtained approximation is still negative.

Pseudo-code for these two approaches is presented in Algorithm 2. To improve the performance of these heuristics, the value of ϵ is doubled at each iteration (but bounded by 0.1). Note that this doubling of ϵ significantly speeds up H2, and is less important for H1.

Figure 3 shows the influence of the chosen number of discretization points, D, and (initial) parameter ϵ for the first heuristic. We observe that the choice of ϵ do not influence performance much. On the other hand, higher-degree polynomials require more discretization points to obtain a more accurate result. However, considering too many discretization points do not improve performance as the error is computed only at the $m = 100$ discretization points $\{\tau_j\}_{j=1}^{100}$. Our second heuristic shows very similar performance, except that it is roughly twice as slow, as it typically requires more inner iterations to converge (around 15 instead of 7).

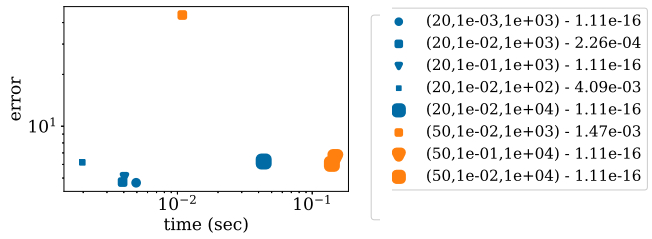


Fig. 3: Influence of the parameters on first heuristic. Different symbols correspond to different values of parameter ϵ (bigger symbols for more discretization points). Legend indicates (d, ϵ, D) - estimated length of negative subintervals.

Algorithm 2: Iterative heuristics for projection

PRE: $\boldsymbol{f}$ is the polynomial to project and $\boldsymbol{\Pi f}$ evaluates it at points $\mathcal{D}$.

Function `HeuriProj` ($\boldsymbol{f}$, $\mathcal{D}$, $\boldsymbol{\Pi}$, $\epsilon > 0$, *maxiter*) **:**

- iter = 0, d = degree of $\boldsymbol{f}$
- vals = $\boldsymbol{\Pi f}$, $\boldsymbol{g} = \boldsymbol{f}$
- **while** *min*($\boldsymbol{\Pi g}$)< 0 *and iter*$<$*maxiter* **do**
  - **if** *H1* **then** `// H1`
    - vals = $\boldsymbol{\Pi g}$
    - vals[vals$<$0] = ϵ
  - **else** `// H2`
    - neg = $\boldsymbol{\Pi g} < 0$
    - vals[neg] = max(vals[neg],0) + ϵ
  - $\boldsymbol{g}$ = `PolynomialCurveFitting`($\mathcal{D}$, vals, d)
  - iter ++, $\epsilon = \min(0.1, \epsilon * 2)$
- **return** $\boldsymbol{g}$

3.2. Proximal projection

Finally, we propose a simple way to approximate the projection of a polynomial for which the projection of a nearby polynomial is already known. Suppose that the projection of $f_1(t)$ is $g_1(t)$, and that $f_2(t)$ is close to $f_1(t)$, with $\delta(t) = f_1(t) - f_2(t)$ and $\boldsymbol{\delta}$ the coefficient vector of $\delta(t)$ ($||\boldsymbol{\delta}||$ small). The problem to solve is:

$$\boldsymbol{g_2} = \text{argmin}_{\boldsymbol{g}} ||\boldsymbol{L}(\boldsymbol{f_1} - \boldsymbol{\delta} - \boldsymbol{g})||_2^2 \quad \text{such that } \boldsymbol{g} \in \mathcal{P}_+^d([a,b])$$

which is equivalent to

$$\boldsymbol{g_2} + \boldsymbol{\delta} = \text{argmin}_{\boldsymbol{g}} ||\boldsymbol{L}(\boldsymbol{f_1} - \boldsymbol{g})||_2^2 \quad \text{s.t. } \boldsymbol{g} \in \mathcal{P}_+^d([a,b]) + \boldsymbol{\delta}$$

This last problem is very similar to the one used to find $\boldsymbol{g_1}$, except that $\boldsymbol{g}$ must belong to $\mathcal{P}_+^d([a,b])+\boldsymbol{\delta}$ instead of $\mathcal{P}_+^d([a,b])$. Nevertheless, as $\delta(t)$ is a small perturbation, we propose to estimate $\boldsymbol{g_2}$ as $\boldsymbol{g_2} = \boldsymbol{g_1} - \boldsymbol{\delta}$. Of course, polynomial $g_2(t)$ obtained in such a way can contain negative values. Therefore, we search for the maximal $\gamma \in [0,1]$ such that $\boldsymbol{g_2} = \boldsymbol{g_1} - \gamma\boldsymbol{\delta}$ is nonnegative over the considered interval. As $\boldsymbol{g_1}$ is nonnegative such a γ always exists and should be as large as possible to stay close to the estimated solution $\boldsymbol{g_1} - \boldsymbol{\delta}$. Since $\mathcal{P}_+^d([a,b])$ is a convex set, we can use a bisection search to find the maximal γ (nonnegativity of $\boldsymbol{g_2}$ is checked on a set $\mathcal{D}$ of D discretization points defined as for the other heuristics).

Figure 4 illustrates performance of the proximal algorithm. In this case, increasing the number of discretization points do not reduce the error too much but increases the needed time, and decreases the length of negative subintervals. As expected, the algorithm performs better when the known polynomial is closer to the polynomial to project.

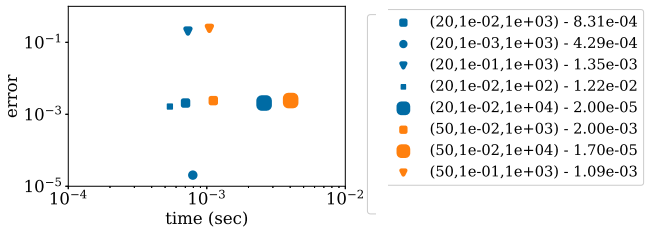
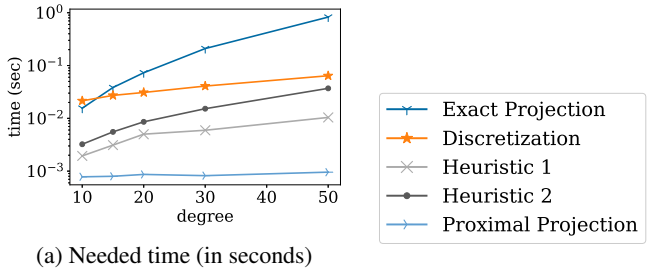


Fig. 4: Influence of the parameters on proximal heuristic. Different symbols correspond to different distances between the polynomial to project and the polynomial with known projection, bigger symbols for more discretization points. Legend indicates (d, $||\delta||$, D) - estimated length of negative subintervals.

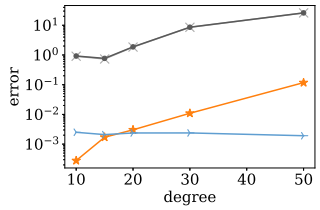
3.3. Performance comparison for heuristics

Figure 5 show performances of each heuristic in comparison to the exact projection. The polynomial to project is evaluated at 100 equispaced points, while set $\mathcal{D}$ contains 10^3 equispaced points and length of negative subintervals is estimated trough 10^6 points. Parameter ϵ is chosen as 10^{-2} for heuristics 1 and 2. To evaluate the proximal projection, we use a polynomial with known projection and impose the coefficient vector of this polynomial $\boldsymbol{f_1}$ to be at distance 10^{-2} from $\boldsymbol{f}$, the coefficient vector of the polynomial to project ($||\boldsymbol{f_1} - \boldsymbol{f}||_2 = 10^{-2}$).

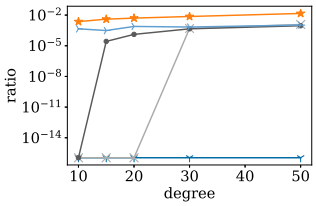
Figure 5(a) shows that the heuristics are significantly faster than the exact projection, especially for large degrees. Proximal projection is the fastest, but requires to know the



(a) Needed time (in seconds)



(b) Projection error



(c) Ratio of negative points

Fig. 5: Performances of the heuristics presented in this section.

projection of a close polynomial. The two iterative heuristics display similar performances, even though heuristic 2 is slightly slower, and are faster than the discretization heuristic. In Figure 5(b), we observe that their approximation error is similar, and much larger than the error of the two other approaches. Figure 5(c) illustrates that all methods lead to a solution with only very few negative points (on average less than 2% of the total interval length for all methods). The discretization heuristic performs worst on that aspect.

We conclude that if information about close polynomials is known, using the proximal projection leads to very good results almost instantly. Otherwise, the discretization leads to more accurate results but is slower than the two iterative heuristics, which obtain similar results with fewer negative values but also a larger approximation error. The ultimate effect of these approximation errors however will have to be evaluated in the context of the whole algorithm for P-NMF.

4. P-HALS USING APPROXIMATE PROJECTIONS

In this final section we analyze how the use of our heuristics to compute approximate projections impacts the P-HALS algorithm. In particular, we study whether the approximation errors have any negative effect on the convergence of the algorithm and the accuracy of the solutions, and whether the use of faster projections decreases the total computational time.

In our experiments, we choose to use $D = 10^3$ discretization points for all algorithms, parameter ϵ equal to 10^{-2} for heuristics H1 and H2, and decide to apply the proximal projection heuristic when the last updated polynomial is closer than 10^{-1} (which typically happens in later iterations). Degree of the sought basis polynomials is $d = 12$. All algorithms are stopped when iterations no longer improve the cost function: defining cost = $||\boldsymbol{Y} - \boldsymbol{\Pi}\bar{\boldsymbol{B}}\bar{\boldsymbol{X}}||$, we stop when $|\text{cost} - \text{previous cost}|/\text{cost} < 10^{-8}$.

We perform our tests over synthetic input signals, created as $\boldsymbol{Y} = \boldsymbol{A}^*\boldsymbol{X}^* + \boldsymbol{N}$ where matrix $\boldsymbol{A}^*$ contains some ground truth signals, mixing matrix $\boldsymbol{X}^* \in \mathbb{R}^{r\times n}$ is randomly generated using a normal distribution $\mathcal{N}(0,1)$ with negative values replaced by zero, and $\boldsymbol{N} \in \mathbb{R}^{m\times n}$ is an additive Gaussian noise with a signal-to-noise ratio (SNR) of 20 dB. Use of synthetic signals allows us to compute error with respect to ground truth, which is $\boldsymbol{Y}^* = \boldsymbol{A}^*\boldsymbol{X}^*$. If $\bar{\boldsymbol{B}}$ and $\bar{\boldsymbol{X}}$ are the matrices produced by an algorithm, we evaluate accuracy by computing the relative norm of the residual, which is res $= ||\boldsymbol{Y}^* - \boldsymbol{\Pi}\bar{\boldsymbol{B}}\bar{\boldsymbol{X}}||/||\boldsymbol{Y}^*||$.

We now proceed to compare the average performance of several algorithms for P-NMF, including standard HALS, the P-LS method based on least-squares [11] as well as P-HALS with and without approximate projections (using the discretization, H1 and H2 heuristics). The proximal projection heuristic is tested in combination both with P-HALS alone and with P-HALS using H1.

Polynomial input signals. We first test with an input data set containing $r = 3$ polynomials of degree 12 with $n = m = 500$. We observe in table 1 that HALS is fastest but obtains a final solution with significantly worse accuracy than all other methods, as it ignores the polynomial structure of the input signals. The P-LS method is more accurate but much slower than all other methods. All methods based on P-HALS compute even more accurate solutions. Among those, methods using approximate projections are, as expected, faster or much faster. More surprising is the fact that, despite their use of inexact projections, they converge to final solutions with accuracy similar to P-HALS with exact projections, and even sometimes a little better and using fewer iterations. Use of the proximal projection heuristic further decreases computational times, at the cost of very slightly larger residues.

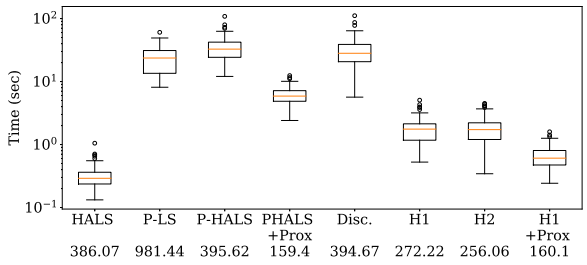
Method	Time	Iterations	Res
HALS	0.66	224.62	0.01437
P-LS	36.1	448.39	0.01075
P-HALS	10.87	186.84	0.00890
P-HALS+Prox	6.94	150.67	0.00896
P-HALS with Discr.	10.56	178.32	0.00891
P-HALS with H1	2.55	170.73	0.00792
P-HALS with H2	4.94	161.13	0.00793
P-HALS with H1+Prox	2.03	151.44	0.00795

Table 1: Average performance on synthetic polynomial signals over 10 tests using 10 initializations (100 tests in total), of HALS, P-LS, P-HALS, and five of our heuristic variants of P-HALS.

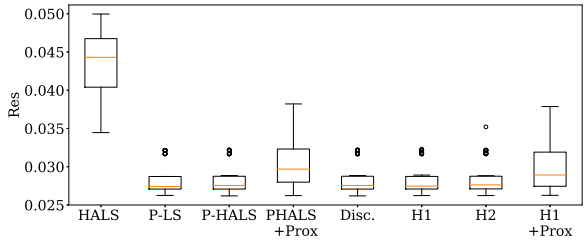
Real reflectance input signals. We also tested our algorithms with $r = 5$ real reflectance signals discretized over $m = 414$ points equally spaced over $[-1, 1]$, coming from the U.S. Geological Survey database [17][1].

[1] Adulania, Clinochlore, Hypersthene, Olivine and Spessatine from https://www.usgs.gov/labs/spec-lab/capabilities/spectral-library

Figure 6 compares performances of the same algorithms, over 100 tests. Again, we see that our heuristics allows P-HALS to converge faster, especially when using heuristics H1 or H2. Using the proximal projection results to even faster computations using fewer iterations, but with slightly larger final residues. This may be due to an early stop of the algorithm because the proximal projection stays too close from previous iterate.



(a) Distribution of CPU time (in seconds) + mean iterations in legend



(b) Final residual (res)

Fig. 6: P-NMF algorithms applied to real reflectance signals.

From the presented tests, we advise to use heuristic H1 with P-HALS as it obtains the best performance and performs better projection than H2, as presented in section 3.3. The use of proximal projection in P-HALS leading to slightly less accurate solutions further accelerate the algorithm.

5. CONCLUSION

Nonnegative matrix factorization problems over polynomial signals (P-NMF) can be successfully tackled using the P-HALS algorithm, which requires repeated projections over sets of nonnegative polynomials on some interval. Computing these projections is computationally expensive, but our numerical experiments suggest that performing an exact projection is not necessary for P-HALS to converge, and show that the heuristic projection algorithms introduced in this paper significantly accelerate the P-HALS algorithm while preserving the final accuracy of the computed solutions.

6. REFERENCES

[1] A. Cichocki, R. Zdunek, A. H. Phan, and S. Amari, *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[2] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, p. 788, 1999.

[3] I. Buciu, N. Nikolaidis, and I. Pitas, "Nonnegative matrix factorization in polynomial feature space," *IEEE Transactions on Neural Networks*, vol. 19, no. 6, pp. 1090–1100, 2008.

[4] A. Darsono, C. Toh, M. M. Saat, A. Isa, N. Manap, and M. Ibrahim, "β-divergence nonnegative matrix factorization on biomedical blind source separation," *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*, vol. 9, no. 2, pp. 1–4, 2017.

[5] P. Sajda, S. Du, T. R. Brown, R. Stoyanova, D. C. Shungu, X. Mao, and L. C. Parra, "Nonnegative matrix factorization for rapid recovery of constituent spectra in magnetic resonance chemical shift imaging of the brain," *IEEE transactions on medical imaging*, vol. 23, no. 12, pp. 1453–1465, 2004.

[6] D. Donoho and V. Stodden, "When does non-negative matrix factorization give a correct decomposition into parts?" in *Advances in Neural Information Processing Systems 16*. MIT Press, 2004, pp. 1141–1148.

[7] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of machine learning research*, vol. 5, no. Nov, pp. 1457–1469, 2004.

[8] S. Choi, "Algorithms for orthogonal nonnegative matrix factorization," *Neural Networks IJCNN*, pp. 1828–1832, 2008.

[9] R. Zdunek, "Alternating direction method for approximating smooth feature vectors in nonnegative matrix factorization," in *2014 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2014, pp. 1–6.

[10] R. Zdunek, A. Cichocki, and T. Yokota, "B-spline smoothing of feature vectors in nonnegative matrix factorization," in *International Conference on Artificial Intelligence and Soft Computing*. Springer, 2014, pp. 72–81.

[11] O. Debals, M. Van Barel, and L. De Lathauwer, "Nonnegative matrix factorization using nonnegative polynomial approximations," *IEEE Signal Processing Letters*, vol. 24, no. 7, pp. 948–952, 2017.

[12] D. Backenroth, *Methods in functional data analysis and functional genomics*. Columbia University, 2018.

[13] C. Hautecoeur and F. Glineur, "Nonnegative matrix factorization with polynomial signals via hierarchical alternating least squares," in *European Symposium on Artificial Neural Networks (ESANN)*, 2019, pp. 125–130.

[14] A. Cichocki, R. Zdunek, and S.-i. Amari, "Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization," in *International Conference on Independent Component Analysis and Signal Separation*. Springer, 2007, pp. 169–176.

[15] V. Powers and B. Reznick, "Polynomials that are positive on an interval," *Transactions of the American Mathematical Society*, vol. 352, no. 10, pp. 4677–4692, 2000.

[16] B. Reznick, "Some concrete aspects of Hilbert's 17th problem," *Contemporary mathematics*, vol. 253, pp. 251–272, 2000.

[17] R. Kokaly and al., "USGS spectral library version 7," 2017.