ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

Nonnegative matrix factorization with polynomial signals via hierarchical alternating least squares

Cécile Hautecoeur and François Glineur*

Université catholique de Louvain - CORE and ICTEAM Institute
B-1348 Louvain-la-Neuve - Belgium

Abstract. Nonnegative matrix factorization (NMF) is a widely used tool in data analysis due to its ability to extract significant features from data vectors. Among algorithms developed to solve NMF, hierarchical alternating least squares (HALS) is often used to obtain state-of-the-art results. We generalize HALS to tackle an NMF problem where both input data and features consist of nonnegative polynomial signals. Compared to standard HALS applied to a discretization of the problem, our algorithm is able to recover smoother features, with a computational time growing moderately with the number of observations compared to existing approaches.

1 Nonnegative matrix factorization with polynomial data

Nonnegative Matrix Factorization (NMF) is a linear dimensionality reduction technique designed to extract characteristic features from data sets of nonnegative vectors [1],[2] using a part-based representation. Besides feature extraction, NMF techniques compress data while filtering the noise. Their performances can be improved using a priori knowledge on the data, such as sparsity [3], smoothness, orthogonality [4], etc.

In this work we assume that input data are continuous nonnegative signals, and we factorize them using linear combinations of low-degree nonnegative polynomials, generalizing classical NMF. The use of low-degree polynomials is motivated by their ability to describe smooth signals and their efficient parametrization (via a list of coefficients in a well-chosen basis).

1.1 Nonnegative matrix factorization (NMF)

Given a data matrix $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$, the goal of NMF is to recover matrices $\boldsymbol{A} \in \mathbb{R}_+^{m\times r}$ and $\boldsymbol{X} \in \mathbb{R}_+^{r\times n}$ such that $\boldsymbol{Y} \approx \boldsymbol{AX}$, where rank r of the approximation is typically much smaller than m and n. Viewing each column of the input $\boldsymbol{Y}$ as an observation $y_{:i}$, NMF expresses each of these n observations as a linear combination (with coefficients in $\boldsymbol{X}$) of r well-chosen common basis elements $a_{:k}$ (columns of $\boldsymbol{A}$). Several cost functions can be considered to measure accuracy of the approximation, and in this work we focus on the Frobenius distance:

$$\min_{\boldsymbol{A}\in\mathbb{R}_+^{m\times r},\boldsymbol{X}\in\mathbb{R}_+^{r\times n}} ||\boldsymbol{Y}-\boldsymbol{AX}||_F^2 = \min_{\substack{a_{:k}\in\mathbb{R}_+^m, x_{ki}\geq 0 \\ \forall\ 1\leq k\leq r, 1\leq i\leq n}} \sum_{i=1}^n ||y_{:i} - \sum_{k=1}^r a_{:k}x_{ki}||_2^2 \ .$$

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

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

This problem, with a non-convex objective function, is proven to be NP-Hard in [5]. It is however convex with respect to $\boldsymbol{A}$ or $\boldsymbol{X}$, hence many NMF algorithms consist of (approximately) solving the problem alternatively on both matrices.

1.2 Hierarchical Alternative Least Squares (HALS)

To perform one iteration of HALS [6], rather than optimizing over the whole matrix $\boldsymbol{A}$, one successively optimizes over each of its columns $a_{:j}$ separately, using the exact closed-form formula below (obtained as the minimizer of a convex quadratic function in $a_{:j}$ projected on the nonnegative orthant). Then a similar update is performed over each row $x_{j:}$ of $\boldsymbol{X}$, before repeating the whole process.

$$a_{:j} = \left[\frac{\boldsymbol{Y}x_{j:}^\top - \sum_{k\neq j} a_{:k}x_{k:}\ x_{j:}^\top}{x_{j:}x_{j:}^\top}\right]_+ \qquad x_{j:} = \left[\frac{a_{:j}^\top\boldsymbol{Y} - a_{:j}^\top\sum_{k\neq j} a_{:k}x_{k:}}{a_{:j}^\top a_{:j}}\right]_+.$$

HALS is frequently used to obtain state-of-the-art results for NMF (see e.g. [7]).

1.3 NMF with polynomial signals (NMF-P) and previous work

In this work, we consider both the input data and the sought basis elements to be nonnegative univariate polynomial signals. Let $\mathcal{P}_+^D(I)$ be the set of all degree D polynomials that are nonnegative on interval I. Assuming that both D and I are known in advance, the NMF problem with polynomial signals (NMF-P) considers that each of the n columns of the input data $\boldsymbol{Y} \in \mathbb{R}^{m\times n}$ is an observation of a nonnegative polynomial $y_i(t) \in \mathcal{P}_+^D$ over m discretization points $\{t_\tau\}_{\tau=1}^m$, potentially with some noise added, i.e. that $y_{\tau,i} = y_i(t_\tau) + n_{\tau,i}$. It is then natural to assume that the basis elements, i.e. the columns of $\boldsymbol{A}$, are also observations of some nonnegative polynomials, hence that $a_{\tau,j} = a_j(t_\tau)$ for some polynomials $a_j(t) \in \mathcal{P}_+^D(I)$. NMF-P consists in computing the r best polynomials $\{a_j(t)\}_{j=1}^r$ and the best nr nonnegative weights $\boldsymbol{X} \in \mathbb{R}_+^{r\times n}$.

NMF-P was recently considered by Debals et al. [8], who propose to solve it as a standard non-linear least-squares problem using well-chosen parametrizations to handle the non-negativity constraints. Parametrizing the nonnegativity conditions on weights $\boldsymbol{X}$ is straightforward with $x_{ij} = h_{ij}^2\ \forall i,j$ (componentwise square). To parametrize matrix $\boldsymbol{A}$, i.e. the coefficients of the nonnegative basis polynomials $a_j(t)$, they use from [9] (note that from here we fix $I = [-1,\ 1]$, as this interval can always be obtained after an appropriate change of variable):

$$f \in \mathcal{P}_+^D([-1,1]) \Leftrightarrow f(t) = f_1(t) + (1-t^2)f_2(t) \qquad f_1 \in \mathcal{P}_+^D(\mathbb{R}),\ f_2 \in \mathcal{P}_+^{D-2}(\mathbb{R}).$$

Moreover, it is well-known that $h \in \mathcal{P}_+^D(\mathbb{R})$ is equivalent to $h(t) = \sum_i (h_i(t))^2$ (called a sum-of-squares (SOS) representation). Actually, based on [9], the authors of [8] state that it is enough to impose f_1 and f_2 to be square polynomials.

This Least-Squares approach, which we will denote LS, does not perform very well for standard NMF (where state-of-the-art methods use alternating schemes [10, 6]). However, for NMF-P, LS reduces the number of unknowns from $mr+nr$ to $(D+1)r + nr$, which is significant when the number of discretization points is large ($m \gg D$) and partially explains the good performance reported in [8].

The present work will instead use an alternating scheme to solve NMF-P.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

2 Two generalizations of HALS to NMF-P

Recall that columns of $\boldsymbol{Y}$ and $\boldsymbol{A}$ are (observations of) polynomials in $\mathcal{P}_+^D([-1,1])$. For simplicity, we denote them as $y_i(t)$ and $a_j(t)$, $t \in [-1,1]$. We can either consider that the coefficients of polynomials y_i are known (or obtained via regression), or that only evaluations at m discretized times $\{\tau_j\}_{j=1}^m$ are accessible, leading to two different cost functions to minimize during each HALS iteration:

$$\sum_{i=1}^{n} \int_{-1}^{1} \big(y_i(t) - \textstyle\sum_{k=1}^{r} a_k(t) x_{k,i}\big)^2 \, dt \quad \text{and} \quad \sum_{i=1}^{n} \sum_{j=1}^{m} \big(y_i(\tau_j) - \textstyle\sum_{k=1}^{r} a_k(\tau_j) x_{k,i}\big)^2.$$

coefficients known, integral case — only observations, sum case

2.1 Integral case: PI-HALS (coefficients of input polynomials known)

Evaluating a polynomial at t can be written as $p(t) = \boldsymbol{\pi}(t)\boldsymbol{p}$ where $\boldsymbol{p}$ is a column vector of coefficients in a fixed basis of polynomials (e.g. monomials), and $\boldsymbol{\pi}(t)$ is a row vector containing these basis polynomials evaluated at t. Introducing residuals $e_i(t) = y_i(t) - \sum_{k=1}^{r} a_k(t) x_{k,i}$, we define the following (quadratic) cost:

$$\sum_{i=1}^{n} \int_{-1}^{1} e_i(t)^2 dt = \sum_{i=1}^{n} \int_{-1}^{1} (\boldsymbol{\pi}(t)\boldsymbol{e_i})^2 dt = \sum_{i=1}^{n} \boldsymbol{e_i}^\top M \boldsymbol{e_i} \text{ with } M = \int_{-1}^{1} \boldsymbol{\pi}(t)^\top \boldsymbol{\pi}(t) dt.$$

Letting the matrices of coefficients of $\boldsymbol{Y}$ and $\boldsymbol{A}$ respectively be $\boldsymbol{Z} \in \mathbb{R}^{(D+1)\times n}$ and $\boldsymbol{B} \in \mathbb{R}^{(D+1)\times r}$ (with columns $b_{:j}$), the derivatives of this cost C are:

$$\frac{\partial C}{\partial b_{:j}} = -2M\big(\boldsymbol{Z}x_{j:}^\top - \textstyle\sum_{k=1}^{r} b_{:k} x_{k:} x_{j:}^\top\big) \qquad \frac{\partial C}{\partial x_{j:}} = -2\big(b_{:j}^\top M \boldsymbol{Z} - \textstyle\sum_{k=1}^{r} b_{:j}^\top M b_{:k} x_{k:}\big)$$

which produces the following generalized HALS updates for NMF-P:

$$b_{:j} = \left[\frac{\boldsymbol{Z}x_{j:}^\top - \sum_{k\neq j} b_{:k} x_{k:}\, x_{j:}^\top}{x_{j:} x_{j:}^\top}\right]_{\mathcal{P}_+^D([-1,1])} \qquad x_{j:} = \left[\frac{b_{:j}^\top M \boldsymbol{Z} - \sum_{k\neq j} b_{:j}^\top M b_{:k} x_{k:}}{b_{:j}^\top M b_{:j}}\right]_+.$$

Updating the rows $x_{j:}$ of $\boldsymbol{X}$ is easy and similar to the usual HALS. For the polynomials in $\boldsymbol{A}$ (i.e. their coefficients $b_{:j}$), we need to compute a projection onto the set of non-negative polynomials, which is not straightforward. We rely on the sum-of-squares approach described in section 1.3 and compute the projection of a given polynomial $p(t) = \boldsymbol{\pi}(t)\boldsymbol{p}$ as a (convex) quadratic minimization problem over $x(t) = \boldsymbol{\pi}(t)\boldsymbol{x}$ parameterized by two SOS polynomials $q(t)$ and $r(t)$:

$$\min_{\boldsymbol{x}} (\boldsymbol{x}-\boldsymbol{p})^\top M (\boldsymbol{x}-\boldsymbol{p}) \quad \text{such that } x(t) = q(t) + (1-t^2) r(t);\ q(t), r(t) \in SOS.$$

As the set of univariate sum-of-squares polynomials is a convex cone that can be represented with a linear matrix inequality [11], our projection can be computed exactly and efficiently as the solution of a semidefinite optimization problem (we used the MOSEK 8.1 solver). To the best of our knowledge this projection onto nonnegative univariate polynomials has not been studied before. Figure 1 illustrates such a projection.

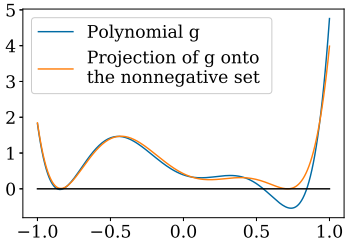

Fig. 1: Example of projection onto the set of nonnegative polynomials.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

2.2 Sum case: PS-HALS (input polynomials known via observations)

Updates for this case are similar to the original HALS presented in section 1.2, with the difference that matrix $\boldsymbol{A}$ is parameterized as $\boldsymbol{A} = \boldsymbol{\Pi}\boldsymbol{B}$ where $\boldsymbol{\Pi} \in \mathbb{R}^{m\times(D+1)}$ is a fixed matrix (made of m rows $\boldsymbol{\pi}(\tau_j)$ stacked vertically), and $\boldsymbol{B} \in \mathbb{R}^{(D+1)\times r}$ is the unknown matrix of polynomial coefficients. Using pseudo-inverse $\boldsymbol{\Pi}^\dagger = (\boldsymbol{\Pi}^\top\boldsymbol{\Pi})^{-1}\boldsymbol{\Pi}^\top$, $\hat{\boldsymbol{\Pi}} = \boldsymbol{\Pi}^\top\boldsymbol{\Pi}$ and $\mathcal{P}_1 = \mathcal{P}_+^D([-1,1])$, we have:

$$b_{:j} = \left[\frac{\boldsymbol{\Pi}^\dagger \boldsymbol{Y} x_{j:}^\top - \sum_{k\neq j} b_{:k}x_{k:}\, x_{j:}^\top}{x_{j:}x_{j:}^\top}\right]_{\mathcal{P}_1} \qquad x_{j:} = \left[\frac{b_{:j}^\top\boldsymbol{\Pi}^\top\boldsymbol{Y} - b_{:j}^\top\hat{\boldsymbol{\Pi}}\sum_{k\neq j} b_{:k}x_{k:}}{b_{:j}^\top\hat{\boldsymbol{\Pi}}b_{:j}}\right]_+.$$

3 Preliminary numerical results

We perform a preliminary performance study of our algorithms PI-HALS and PS-HALS, specifically designed to handle NMF-P with polynomial signals, and compare them to standard HALS and LS. Synthetic data is used: to create $\boldsymbol{Y}$ we randomly select r nonnegative polynomials on $[-1,1]$ and evaluate them on m points to form matrix $\boldsymbol{A}$, while matrix $\boldsymbol{X}$ contains random nonnegative numbers. We then create the data $\boldsymbol{Y} = \boldsymbol{A}\boldsymbol{X} + \boldsymbol{N}$ where $\boldsymbol{N} \in \mathbb{R}^{m\times n}$ contains additive Gaussian noise with a given SNR. As our PI-HALS algorithm requires the coefficients of the polynomials in $\boldsymbol{Y}$, these are estimated via regression.
In our experiments, we use a Chebyshev basis to represent polynomials and, when applying HALS (for both the integral and the sum case), we update matrix $\boldsymbol{X}$ several times before alternating, as suggested in [7].

3.1 Quality of the recovery of polynomial signals

We first study how well the original polynomial signals are recovered; we choose here $m = n = 100$, $r = 4$ and $D = 6$. Figure 2 (left) shows that in the noiseless case, all algorithms are able to perfectly recover the original polynomials[1] in $\boldsymbol{A}$.

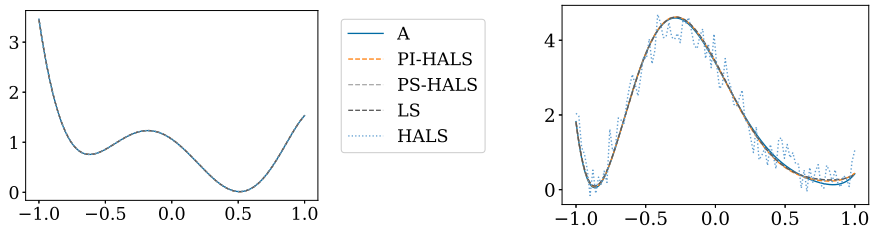

Fig. 2: Example of recovered polynomials. Left: noise-free case, Right: noisy case.

In the noisy case (Figure 2, right), with a signal-to-noise ratio (SNR) equal to 10 dB, features recovered by polynomial-based algorithms (PI-HALS, PS-HALS and LS) are all very close to the true signals, and much smoother than when original HALS is used, demonstrating the usefulness of NMF-P.

[1]Since a factorization is defined up to permutations and scaling of the columns of $\boldsymbol{A}$ and rows of $\boldsymbol{X}$, some care is needed when considering a candidate solution $\bar{\boldsymbol{A}}$: we observe the matrix $\bar{\boldsymbol{A}} = \boldsymbol{A_1}\boldsymbol{Q}$ where $\boldsymbol{A_1}$ is the recovered matrix and $\boldsymbol{Q} \in \mathbb{R}^{r\times r}$ minimizes $||\boldsymbol{A_1}\boldsymbol{Q} - \boldsymbol{A}||$.

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0.
Available from http://www.i6doc.com/en/.

3.2 Comparison with the least-square approach

Figure 3 below displays the evolution of the approximation error after each iteration, using the same noisy data as above. We report the 'true' error computed w.r.t. the original polynomials used to create data (before noise was added). Costs for (PS-HALS), (HALS) and (LS) were scaled by a factor $\frac{2}{m}$ to become comparable with the integral used by (PI-HALS).

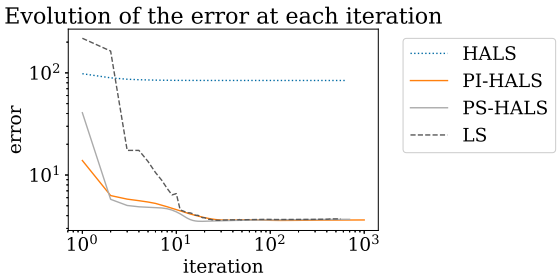

Fig. 3: Evolution of the error after each iteration.

We observe that the three polynomial-based methods have similar results on the original data, while HALS leads to an higher error. This observation is confirmed when the algorithms are run several times on different problems. Figure 4 illustrates the mean approximation error obtained when the number of discretization points m increases (left) or when the number of observations n increases (right), with $D = 12$, $r = 4$ and $SNR = 20$ dB. We observe that polynomial-based method are better than HALS in general but especially when m is high or n is low.

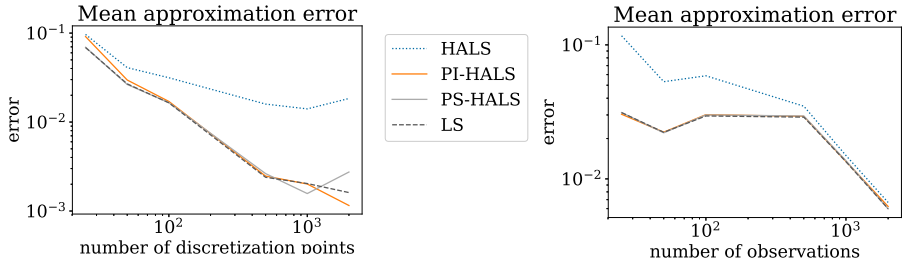

Fig. 4: Error for increasing m (left) and n (right). Mean over 10 runs/10 initializations.

To study computational performance we now run each algorithm during 20 iterations for an increasing number of observations and discretization points ($m = n$), with $D = 12$, $r = 4$ and $SNR = 20$ dB. Figure 5 illustrates that one iteration of HALS has $\mathcal{O}(mn)$ complexity, while for LS it is only $\mathcal{O}(n)$ [8]. For our methods, on the other hand, time spent in computations increases

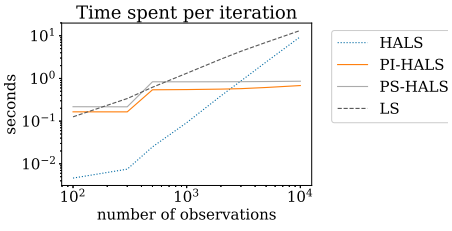

Fig. 5: Spent time for increasing $m = n$ (mean over 10 runs/10 initializations).

ESANN 2019 proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 24-26 April 2019, i6doc.com publ., ISBN 978-287-587-065-0. Available from http://www.i6doc.com/en/.

very moderately with $m = n$, because our algorithms spend a large fraction of their time on the projection step, which does not depend on m, neither n.

4 Conclusion and discussion

We have shown that extending NMF to handle polynomial signals directly enables recovery of smoother features and is less sensitive to noise than standard HALS applied to discretized signals.

We have presented two new algorithms to perform NMF with polynomial signals, and showed that they provide accurate results. Another advantage compared to existing methods is that their computation times only increase moderately with the problem size, making them more interesting to deal with large-scale problems (i.e. with large numbers of observations or discretization points) compared to the previously proposed least-squares approach from [8].

Our algorithms are naturally well-suited to deal with data originating from nonnegative polynomials. Adapting the methods to other nonnegative interpolating functions, such as splines, would be an interesting topic for future research.

References

[1] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788, 1999.

[2] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. *Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation*. John Wiley & Sons, 2009.

[3] Patrik O Hoyer. Non-negative matrix factorization with sparseness constraints. *Journal of machine learning research*, 5(Nov):1457–1469, 2004.

[4] Seungjin Choi. Algorithms for orthogonal nonnegative matrix factorization. *Neural Networks IJCNN*, pages 1828–1832, 2008.

[5] Stephen A Vavasis. On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, 20(3):1364–1377, 2009.

[6] Andrzej Cichocki, Rafal Zdunek, and Shun-ichi Amari. Hierarchical ALS algorithms for nonnegative matrix and 3d tensor factorization. In *International Conference on Independent Component Analysis and Signal Separation*, pages 169–176. Springer, 2007.

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

[8] Otto Debals, Marc Van Barel, and Lieven De Lathauwer. Nonnegative matrix factorization using nonnegative polynomial approximations. *IEEE Signal Processing Letters*, 24(7):948–952, 2017.

[9] Victoria Powers and Bruce Reznick. Polynomials that are positive on an interval. *Transactions of the American Mathematical Society*, 352(10):4677–4692, 2000.

[10] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. *SIAM journal on matrix analysis and applications*, 30(2):713–730, 2008.

[11] Grigoriy Blekherman, Pablo A Parrilo, and Rekha R Thomas. *Semidefinite optimization and convex algebraic geometry*. SIAM, 2012.