LETTER **Communicated by Xiao Fu**

Accelerating Nonnegative Matrix Factorization Algorithms Using Extrapolation

Andersen Man Shun Ang
manshun.ang@umons.ac.be
Nicolas Gillis
nicolas.gillis@umons.ac.be
Department of Mathematics and Operational Research, Faculté Polytechnique, Université de Mons, 7000 Mons, Belgium

We propose a general framework to accelerate significantly the algorithms for nonnegative matrix factorization (NMF). This framework is inspired from the extrapolation scheme used to accelerate gradient methods in convex optimization and from the method of parallel tangents. However, the use of extrapolation in the context of the exact coordinate descent algorithms tackling the nonconvex NMF problems is novel. We illustrate the performance of this approach on two state-of-the-art NMF algorithms: accelerated hierarchical alternating least squares and alternating nonnegative least squares, using synthetic, image, and document data sets.

1 Introduction

Given an input data matrix $X \in \mathbb{R}^{m \times n}$ and a factorization rank r, we consider in this letter the following optimization problem:

$$\min_{W \in \mathbb{R}^{m \times r}, H \in \mathbb{R}^{r \times n}} ||X - WH||_F^2 \quad \text{such that} \quad W \geq 0 \text{ and } H \geq 0. \tag{1.1}$$

This problem is referred to as nonnegative matrix factorization (NMF) and has been shown to be useful in many applications, such as image analysis and document classification (Lee & Seung, 1999). Note that there exist many variants of equation (1.1) using other objective functions and additional contraints or penalty terms on W and H (see Cichocki, Zdunek, Phan, & Amari, 2009; Gillis, 2014, 2017; and Fu, Huang, Sidiropoulos, & Ma, 2018, for more details about NMF models and their applications).

1.1 Algorithms for NMF. The focus of this letter is algorithm design for equation 1.1. Almost all algorithms for NMF use a two-block coordinate descent scheme by optimizing alternatively over W for H fixed and vice versa (see algorithm 1). By symmetry, since $||X - WH||_F = ||X^T - H^T W^T||_F$,

Neural Computation **31, 417–439** (2019)
doi:10.1162/neco_a_01157

Algorithm 1: Framework for Most NMF Algorithms.

Require: An input matrix $X \in \mathbb{R}^{m \times n}$, an initialization $W \in \mathbb{R}_+^{m \times r}$, $H \in \mathbb{R}_+^{m \times r}$.

Ensure: An approximate solution (W, H) to NMF.

1: **for** $k = 1, 2, \ldots$ **do**

2: Update W using a NNLS algorithm to minimize $||X - WH||_F^2$ with $W \geq 0$.

3: Update H using an NNLS algorithm to minimize $||X - WH||_F^2$ with $H \geq 0$.

4: **end for**

the updates of W and H are usually based on the same strategy. Looking at the subproblem for H, the following nonnegative least squares (NNLS) problem,

$$\min_{H \geq 0} ||X - WH||_F^2, \tag{1.2}$$

needs to be solved exactly or approximately. The most popular approaches in the NMF community to solve it are multiplicative updates (Lee & Seung, 1999), active-set methods that solve equation 1.2 exactly (Kim & Park, 2008, 2011), projected gradient methods (Lin, 2007; Guan, Tao, Luo, & Yuan, 2012), and exact block coordinate descent (BCD) methods (Cichocki, Zdunek, & Amari, 2007; Cichocki & Phan, 2009; Hsieh & Dhillon, 2011; Gillis & Glineur, 2012; Chow, Wu, & Yin, 2017). Among these approaches, exact BCD schemes have been shown to be the most effective in most situations (Kim, He, & Park, 2014). The reason is that the optimal update of a single row of H, the others being fixed, admits a simple closed-form solution: we have for all k that

$$\begin{aligned}
&\text{argmin}_{H(k,:) \geq 0} ||X - WH||_F^2 \\
&= \max\left(0, \frac{W(:,k)^T (X - \sum_{j \neq k} W(:,j) H(j,:))}{||W(:,k)||_2^2}\right) \\
&= \max\left(0, \frac{W(:,k)^T X - \sum_{j \neq k} \left(W(:,k)^T W(:,j)\right) H(j,:)}{||W(:,k)||_2^2}\right).
\end{aligned}$$

The algorithm using these updates is referred to as hierarchical alternating least squares (HALS) and updates the rows of H and the columns of W in a sequential way (Cichocki et al., 2007; Cichocki & Phan, 2009). HALS has been improved in several ways:

- Selecting the variable to be updated in order to reduce the objective function the most (Gauss-Seidel coordinate descent; Hsieh & Dhillon, 2011).
- Updating the rows of H several times before updating W (and similarly for the columns of W) as the computation of W^TW and W^TX can be reused, which allows a significant acceleration of HALS (Gillis & Glineur, 2012). This variant is referred to as accelerated HALS (A-HALS).
- Using random shuffling instead of the cyclic updates of the rows of H, which leads in general to better performances (Chow et al., 2017). However, when combined with the above strategies to accelerate HALS, we have observed that the improvement is negligible.

More recently, HALS was also accelerated using randomized sampling techniques (Erichson, Mendible, Wihlborn, & Kutz, 2018).

Although the acceleration scheme proposed in this letter can potentially be applied to any NMF algorithm, we focus for simplicity on two algorithms:

1. A-HALS, which is, as already explained, arguably one of the most efficient NMF algorithms.
2. Alternating nonnegative least squares (ANLS), which is algorithm 1 where the NNLS subproblems 1.2 are solved exactly. To solve the NNLS subproblems, we use the active-set method from Kim and Park (2011), one of the most efficient strategy for NNLS (Kim et al., 2014).

1.2 Outline of the Letter. We introduce a general framework to accelerate NMF algorithms. This framework, described in sections 2 and 3, is closely related to the extrapolation scheme usually used in the context of gradient descent methods. We use it here in the context of exact BCD methods applied to NMF. The difficulty in using this scheme is in choosing the tuning parameters in the extrapolation, for which we propose a simple strategy in section 4. We illustrate the effectiveness of this approach on synthetic, image, and document data sets in section 5.

2 Acceleration through Extrapolation

We describe the simple extrapolation scheme that we will use to accelerate NMF algorithms. This scheme takes its roots in the so-called method of parallel tangents, which is closely related to the conjugate gradient method (Luenberger & Ye, 2015), and the accelerated gradient schemes by Nesterov (2013). The idea is the following. Let us consider an optimization scheme

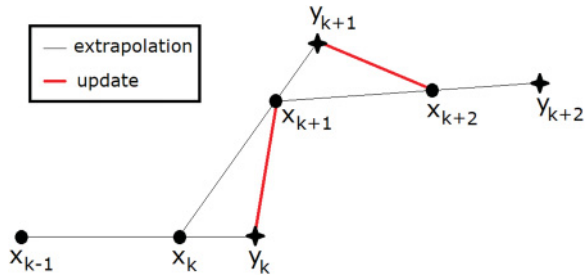

Figure 1: Illustration of the idea of extrapolation to accelerate optimization schemes.

that computes the next iterate only based on the previous iterate[1] (e.g., gradient descent or coordinate descent), that is, it updates the kth iterate x_k as follows:

$$x_{k+1} = \text{update}(x_k),$$

for some function update(.) that depend on the objective function and the feasible set. For most first-order methods, these updates will have a zigzagging behavior. In particular, gradient descent with exact line search leads to orthogonal search directions (Luenberger & Ye, 2015), while search directions of (block) coordinate descent methods are orthogonal by construction. The idea of extrapolation is to define a second sequence of iterates, namely, y_k with $y_0 = x_0$, and modify the above scheme as follows:

$$x_{k+1} = \text{update}(y_k), \quad y_{k+1} = x_{k+1} + \beta_k(x_{k+1} - x_k),$$

for some $\beta_k \geq 0$. Note that there are other possibilities for choosing y_{k+1} based on linear combinations of previous iterates. Figure 1 illustrates the extrapolation scheme and allows us to get some intuition: the direction $(x_{k+1} - x_k)$ will be in between zigzagging directions obtained with the original update applied to y_k's and will allow us to accelerate convergence. For example, we observe in Figure 1 that the direction $x_{k+2} - x_{k+1}$ is between the directions $x_{k+1} - y_k$ and $x_{k+2} - y_{k+1}$.

In the case of gradient descent and smooth convex optimization, the above scheme allows us to accelerate convergence of the function values from $O(1/k)$ to $O(1/k^2)$ and from linear convergence with rate $(1 - \mu/L)$ to rate $(1 - \sqrt{\mu/L})$ for strongly convex function with parameter μ and whose

[1] Although this assumption is not strictly necessary, it makes more sense; otherwise, there might be a countereffect if the update already takes into account the previous iterates.

gradient has Lipschitz constant L (Nesterov, 2013). This scheme has also been used for BCD, and most works focus on the case where the blocks of variables are updated using a gradient or proximal step (see Beck & Tetruashvili, 2013; Xu & Yin, 2013; Fercoq & Richtárik, 2015; Chambolle & Pock, 2015). In the convex case, the β_k's can be chosen a priori in order to obtain the theoretical acceleration. However, from a practical point of view, the acceleration will depend on the choice of the β_k's, which is nontrivial (see Odonoghue & Candès, 2015, for a discussion about this issue). Extrapolation has been used more recently in nonconvex settings (Xu & Yin, 2013; O'Neill & Wright, 2017; Paquette, Lin, Drusvyatskiy, Mairal, & Harchaoui, 2018), but as far as we know, not in combination with exact BCD methods. Xu and Yin (2013) used extrapolation in the context of an inexact BCD method where the blocks of variables are updated using a projected gradient method. Their approach is different from ours, as we will use exact BCD. Note that Xu and Yin (2013) applied their technique to NMF, which we will compare to ours in section 5.

In the method of parallel tangents, the steps β_k are computed using line search (Luenberger & Ye, 2015). This allows the acceleration scheme to be at least as good as the initial scheme. However, this is not a good strategy in our case because the optimal β_k's will be close to zero (because we use coordinate descent). In any case, the choice of the β_k's is nontrivial and, as we will see, the acceleration depends on the choice of these parameters. Note that choosing $\beta_k = 0$ for all k gives back the original algorithm (no extrapolation), and β_k close to one is a very aggressive strategy.

The remainder of this letter is organized as follows:

- In section 3, we adapt the above extrapolation technique in the context of two-block coordinate descent NMF algorithms (see algorithm 1).
- In section 4, we propose a simple strategy for the choice of the parameters β_k's.
- In section 5, we illustrate the acceleration of NMF algorithms on synthetic, image, and document data sets.

3 Extrapolation for NMF Algorithms

In this letter, we adapt extrapolation to the two-block coordinate descent strategies of NMF algorithms described in algorithm 1. Algorithm 2 describes the proposed extrapolation scheme applied to NMF (see equation 1.1). Depending on the choice of the parameter $hp \in \{1, 2, 3\}$, algorithm 2 corresponds to three different variants of the proposed extrapolation. We describe this through two important questions.

3.1 When Should We Perform the Extrapolation? In case of NMF (and in general for BCD methods), it makes sense to perform the extrapolation

Algorithm 2: Acceleration of Algorithm 1 Using Extrapolation.

Require: An input matrix $X \in \mathbb{R}^{m \times n}$, an initialization $W \in \mathbb{R}^{m \times r}_+$ and $H \in \mathbb{R}^{m \times r}_+$, parameters $hp \in \{1, 2, 3\}$ (extrapolation/projection of H).

Ensure: An approximate solution (W, H) to NMF.

$W_y = W$; $H_y = H$; $e(0) = ||X - WH||_F$.
for $k = 1, 2, \ldots$ **do**
    Compute H_n using a NNLS algorithm to minimize $||X - W_y H_n||_F^2$ with $H_n \geq 0$ using H_y as the initial iterate.
    if $hp \geq 2$ **then**
        Extrapolate: $H_y = H_n + \beta_k(H_n - H)$.
    end if
    if $hp = 3$ **then**
        Project: $H_y = \max(0, H_y)$.
    end if
    Compute W_n using a NNLS algorithm to minimize $||X - W_n H_y||_F^2$ with $W_n \geq 0$ using W_y as the initial iterate.
    Extrapolate: $W_y = W_n + \beta_k(W_n - W)$.
    if $hp = 1$ **then**
        Extrapolate: $H_y = H_n + \beta_k(H_n - H)$.
    end if
    Compute the error: $\mathrm{e}(k) = ||X - W_n H_y||_F$. % See Remark 1.
    if $\mathrm{e}(k) > \mathrm{e}(k-1)$ **then**
        Restart: $H_y = H$; $W_y = W$.
        Decrease β_k according to algorithm 3.
    else
        $H = H_n$; $W = W_n$.
        Increase β_k according to algorithm 3.
    end if
end for

scheme after the update of each block of variables so that when we update the next block of variables, the algorithm takes into account the already extrapolated variables (see Fercoq & Richtárik, 2015). However, as we will see in the numerical experiments, this does not necessarily performs best in all cases. This is the first reason that we have added a parameter $hp \in \{1, 2, 3\}$: For $hp = 1$, H is extrapolated after the update of W; otherwise it is extrapolated directly after it has been updated. Note that in the former case, the extrapolated matrix H_y is used only as a warm start for the next NNLS update of H. For ANLS, it will therefore not play a crucial role since ANLS solves the NNLS subproblem exactly.

3.2 Can We Guarantee Convergence? Under some mild assumptions or slight modifications of the algorithm, BCD schemes are guaranteed to converge to stationary points (Hong, Wang, Razaviyayn, & Luo, 2017). Since algorithm 2 uses extrapolation, we cannot use these results directly.

Similarly, we cannot use the result of Xu and Yin (2013) with its projected gradient steps to update W and H. In algorithm 2, because W_y and H_y are not necessarily nonnegative, the objective function is not guaranteed to decrease at each step. In fact, step 16 of algorithm 2 only checks the decrease of $||X - W_n H_y||_F$ where (W_n, H_y) is not necessarily feasible for $hp \geq 2$. The reason for computing $||X - W_n H_y||_F$ and not $||X - W_n H_n||_F$ is threefold. First, W_n was updated according to H_y. Second, it gives the algorithm some degrees of freedom to possibly increase the objective function in the hope of being able to decrease it significantly later. In fact, we have observed in our numerical experiments that this choice allows a faster convergence than when restarting the algorithm based on the error $||X - W_n H_n||_F$. Third, it is computationally cheaper because computing $||X - W_n H_n||_F$ would require $O(mnr)$ operations instead of $O(mr^2)$ (see remark 1).

In order to guarantee the objective function to decrease, a possible way is to require H_y to be nonnegative by projecting it to the nonnegative orthant; this variant corresponds to $hp = 3$. In that case, the solution (W_n, H_y) is a feasible one for which the objective function is guaranteed to decrease at least every second step. In fact, when the error increases, algorithm 2 reinitializes the extrapolation sequence (W_y, H_y) using (W, H) (step 17 of algorithm 2), and the next step is a standard NNLS update. Therefore, since the objective function is bounded below, there exists a converging subsequence of the iterates. Proving convergence to stationary points is an open problem and an important direction for further research. We believe it would be particularly interesting to investigate the convergence of the extrapolation scheme applied on exact BCD in the nonconvex case.

To summarize, using the extrapolation of H after the update of W ($hp = 1$) or using the projection of H_y onto the feasible set ($hp = 3$) is more conservative but guarantees the objective function to decrease (at least every second step). As we will see in the numerical experiments, these two variants perform in general better than with $hp = 2$.

Remark 1 (computation of the error). To compute the error $||X - W_n H_y||_F^2$ in step 15 of algorithm 2 (and in step 1), it is important to take advantage of previous computations and not compute $W_n H_y$ explicitly (which would be impractical for large and sparse matrices). For simplicity, we denote $W = W_n$ and $H = H_y$. We have

$$\begin{aligned} ||X - WH||_F^2 &= \langle X, X\rangle - 2\langle X, WH\rangle + \langle WH, WH\rangle \\ &= ||X||_F^2 - 2\langle W, XH^T\rangle + \langle W^T W, HH^T\rangle. \end{aligned}$$

The term $||X||_F^2$ can be computed once, the term $\langle W, XH^T\rangle$ can be computed in $O(mr)$ operations since MH^T is computed within the NNLS update of W, and the term $\langle W^T W, HH^T\rangle$ requires $O(mr^2)$ since HH^T is also computed within the NNLS update of W. In fact, all algorithms for NNLS we know of

need to compute XH^T and HH^T when solving for W because the gradient of $||X - WH||_F^2$ with respect to W is $2(WHH^T - XH^T)$.

Remark 2 (other NMF variants). Although we focus in this letter on the most standard NMF model, equation 1.1, the acceleration scheme described in algorithm 2 can be directly applied to any NMF model. The only modification to bring to the algorithm is in the way the matrices W and H are updated in steps 3 and 10. For example, one of the most widely used regularizations in NMF is to add ℓ_1 norm penalty terms on W or H in the objective function (i.e., $||W||_1$ and $||H||_1$), to enhance sparsity of the factors (Kim & Park, 2008). In that case, both HALS and ANLS extend directly because the subproblems in variables W and H are still NNLS (in fact, W and H are nonnegative; hence $||W||_1$ and $||H||_1$ are linear terms). Another important example is to look for a matrix W whose columns have minimum volume (Fu et al., 2018) for which Fu, Huang, Yang, Ma, and Sidiropoulos (2016) used a similar extrapolation technique as Xu and Yin (2013) to accelerate significantly their inexact BCD algorithm.

4 Choice of the Extrapolation Parameters β_k's

In this section, we propose a strategy to choose the β_k's. First, we explain why it does not work well to use line search. We focus on the update of W (a similar argument holds for H). We have

$$W_y = W_y(\beta) = W_n + \beta(W_n - W),$$

where W_n is an approximate solution of $\min_{W \geq 0} ||X - WH_y||_F$ (in the case of ANLS, it is an optimal solution). The optimal β can be computed in close form as follows:

$$\beta^* = \text{argmin}_\beta ||X - W_y(\beta)H_y||_F^2 = \frac{\langle X - W_nH_y, (W_n - W)H_y \rangle}{||(W_n - W)H_y||_F^2}.$$

We have observed that β^* is close to zero for most steps of algorithm 2 (β^* is not always close to zero, even when using ANLS, because W_y is not necessarily nonnegative), especially when the algorithm has performed several iterations and reached the neighborhood of a stationary point. The reason is that W_n was optimized to minimize the objective function. Hence, in the following, we propose another strategy to choose the β_k's. It will increase the objective function in most cases (i.e., $||X - W_y(\beta)H_y||_F^2 > ||X - W_y(0)H_y||_F^2$) but will allow a larger decrease of the objective function at the next step. Note that this is the reason why we check whether the error has decreased only after the update of H because otherwise, the acceleration would not be possible (only a small β would be allowed in that case).

Algorithm 3: Update of the β_k's.

Require: Parameters $1 < \bar{\gamma} < \gamma < \eta$, $\beta_1 \in (0, 1)$.

Set $\bar{\beta} = 1$.

if the error decreases at iteration k **then**

    Increase β_{k+1}: $\beta_{k+1} = \min(\bar{\beta}, \gamma\beta_k)$.

    Increase $\bar{\beta}$: $\bar{\beta} = \min\left(1, \bar{\gamma}\bar{\beta}\right)$.

else

    Decrease β_{k+1}: $\beta_{k+1} = \beta_{k+1} = \beta_k/\eta$.

    Set $\bar{\beta} = \beta_{k-1}$.

end if

4.1 Strategy for Updating the β_k's. In this letter, since we are applying the extrapolation scheme to a nonconvex problem using coordinate descent, there is, as far as we know, no a priori theoretically sound choice for the β_k's. For this reason, we consider a very simple scheme described in algorithm 3.

It works as follows. Let us assume there exists a hidden optimal value for the β_k's, as in the strongly convex case where β_k should ideally be equal to $\frac{1-\sqrt{\mu/L}}{1+\sqrt{\mu/L}}$ (Nesterov, 2013; Odonoghue & Candès, 2015), where μ is the strong convexity parameter of the objective function and L is the Lipschitz constant of its gradient. It starts with an initial value of $\beta_0 \in [0, \bar{\beta}]$ and an upper bound $\bar{\beta} = 1$. As long as the error decreases, it increases the value of β_{k+1} by a factor γ, taking into account the upper bound, that is, $\beta_{k+1} = \min(\gamma\beta_k, \bar{\beta})$. It also increases the upper bound by a factor $\bar{\gamma} < \gamma$ if it is smaller than one, that is, $\bar{\beta} = \min(\gamma\bar{\beta}, 1)$. The usefulness of $\bar{\beta}$ is to keep in memory the last value of β_k that allowed a decrease of the objective function, which is used as an upper bound for β_k. However, because the landscape of the objective function may change, $\bar{\beta}$ is slightly increased by a factor $\bar{\gamma} < \gamma$ at each step, as long as the error decreases. When the error increases, β_{k+1} is reduced by a factor $\eta > \gamma$, and the upper bound $\bar{\beta}$ is set to the previous value of β that allowed decrease, that is, β_{k-1}.

Remark 3. We have also tried to mimic the choice of the β_k's from convex optimization (Nesterov, 2013), but in general, it performed worse than the simple choice presented here.

Table 1: Image Data Sets.

Name	Number of Pixels	m	n	r
ORL[a]	112×92	10304	400	40
Umist[b]	112×92	10304	575	40
CBCL[c]	19×19	361	2429	40
Frey[b]	28×20	560	1965	40

[a]http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html.
[b]http://www.cs.toronto.edu/~roweis/data.html.
[c]http://cbcl.mit.edu/software-datasets/FaceData2.html.

Table 2: Text Mining Data Sets from Zhong and Ghosh (2005).

Name	m	n	r	Number of Nonzero	Sparsity
Classic	7094	41,681	20	223,839	99.92
Sports	8580	14,870	20	1,091,723	99.14
Reviews	4069	18,483	20	758,635	98.99
Hitech	2301	10,080	20	331,373	98.57
Ohscal	11,162	11,465	20	674,365	99.47
la1	3204	31,472	20	484,024	99.52

Note: Sparsity is given in %: $100 * \#\text{zeros}/(mn)$).

5 Numerical Experiments

In this section, we show the efficiency of the extrapolation scheme, that is, algorithm 2, to accelerate the NMF algorithms ANLS and A-HALS. All tests are preformed using Matlab R2015a on a laptop Intel CORE i7-7500U CPU at 2.9 GHz 24 GB RAM. (The code is available from https://sites.google.com/site/nicolasgillis/code.)

5.1 Data Sets. We will use the same data sets as in Gillis and Glineur (2012) because they are among the most widely used ones in the NMF literature (see Tables 1 and 2). The image data sets represent facial images and are dense matrices. The document data sets are sparse matrices.

We also consider two types of synthetic data sets. For the first one, which we refer to as the low-rank synthetic data set, we generate each entry of W and H using the uniform distribution in $[0, 1]$ and compute $X = WH$. For each experiment, we generate 10 such matrices and report the average results. For the second one, which we refer to as the full-rank synthetic data set, we simply generate each entry of X uniformly at random in [0,1] so that X is a full rank matrix. In both cases, we use $m = n = 200$ and $r = 20$.

5.2 Experimental Setup. In all cases, we report the average error over 10 random initializations, where the entries of the initial matrices W and H are chosen uniformly at random in the interval $[0, 1]$. To compare the solutions generated by the different algorithms, we follow the strategy from Gillis and Glineur (2012): we report the relative error to which we subtract the lowest relative error obtained by any algorithm with any initialization (denoted $e_{\min}$). Mathematically, given the solution $(W^{(k)}, H^{(k)})$ obtained at the kth iteration, we report

$$E(k) = \frac{||X - W^{(k)}H^{(k)}||_F}{||X||_F} - e_{\min}. \tag{5.1}$$

For the low-rank synthetic data sets, we use $e_{\min} = 0$.

Using $E(k)$ instead of $||X - W^{(k)}H^{(k)}||_F$ has some advantages: (1) it allows meaningfully taking the average results over several data sets, and (2) it provides a better visualization in terms of both initial convergence and the quality of the final solutions computed by the different algorithms. The reason is that $E(k)$ converges to zero for the algorithm that was able to compute the best solution, which allows us to use a logarithmic scale.

5.3 Tuning Parameters: Preliminary Numerical Experiments. Before we compare the two NMF algorithms (ANLS and A-HALS) and their extrapolated variants, we run some preliminary numerical experiments in order to choose reasonable values for the parameter of algorithm 2 (hp) and the parameters to update β_k.

As we will see, the extrapolation scheme performs rather differently for ANLS (it computes an optimal solution of the subproblems) and A-HALS (it computes an approximate solution using a few steps of coordinate descent). It also performs rather differently depending on the value of hp, and it is less sensitive to the values of β_0, γ, $\bar{\gamma}$, and η as long as these values are chosen in a reasonable range.

In the next section, we run the different variants with the following parameters: $\beta_0 = 0.25, 0.5, 0.75$, $\eta = 1.5, 2, 3$, $(\gamma, \bar{\gamma}) = (1.01, 1.005), (1.05, 1.01), (1.1, 1.05)$. For each experiment, we will not be able to display the curve for each extrapolated variant (there would be too many—82 in total: 3^4 and the original algorithm). Therefore, for each value of hp, we display only the variant corresponding to the parameters that obtained the smallest final average error (best) and the largest final average error (worst). This will be interesting to observe the sensitivity of algorithm 2 to the way β_k is updated.

5.3.1 Extrapolated ANLS (E-ANLS). The top two plots of Figure 2 show the evolution of the average of the error measure defined in equation 5.1 for the low-rank and full-rank synthetic data sets. We observe the following:

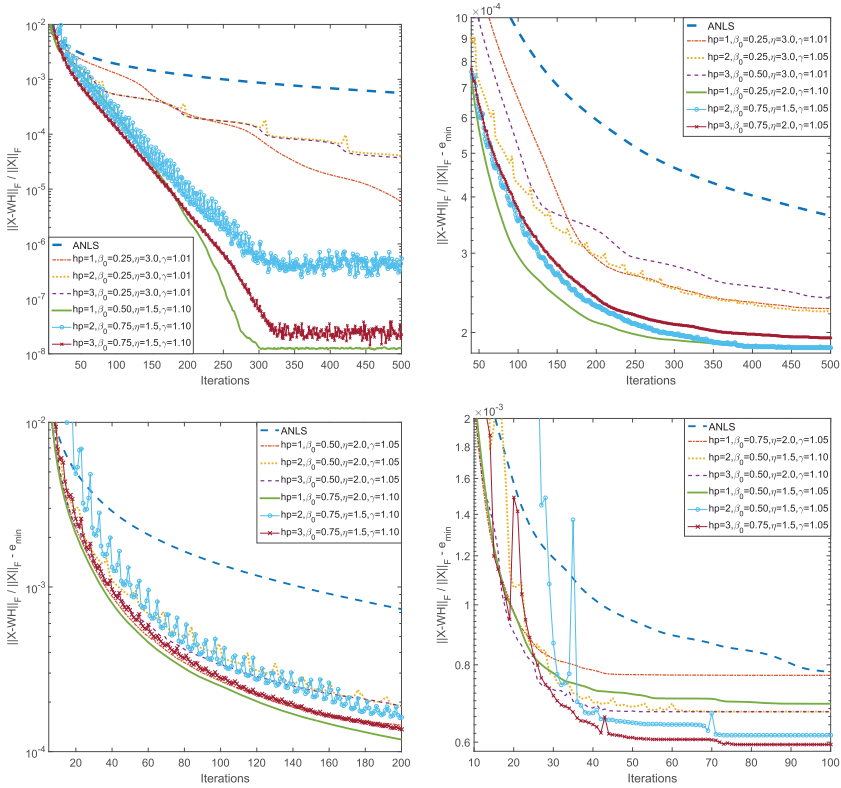

Figure 2: Extrapolation scheme applied to ANLS on the low-rank (top left) and full-rank (top right) synthetic data sets and the image (bottom left) and document (bottom right) real data sets. For each value of hp, we display the corresponding best- and worst-performing variant. The curves are the average value of equation 5.1 among the different data sets and initializations.

- For all the values of the parameters, E-ANLS outperforms ANLS.
- For the low-rank synthetic data sets, E-ANLS with $hp = 1$ and well-chosen parameters for the update of β_k (e.g., $\beta_0 = 0.5$, $\eta = 1.5$, $\gamma = 1.1$) performs extremely well and is able to identify solutions with very small relative error ($\approx 10^{-8}$ in average). In fact, the original ANLS algorithm would not be able to compute such solutions even within several thousand iterations.
- For the full-rank synthetic data sets, E-ANLS variants with hp equal to 1, 2, or 3 perform similarly, although choosing $hp = 1$ allows a slightly faster initial convergence.

- The best value for γ is either 1.05 or 1.10. The best value for η is either 1.5 or 2.0 (3.0 is always the worst). The algorithm is not too sensitive to the initial β, as it is quickly modified within the iterations, but $\beta_0 = 0.25$ clearly provides the worst performance in most cases.

We now perform the same experiment on image and document data sets except with fewer parameters (we do not use $\gamma = 1.01$, $\eta = 3$, $\beta_0 = 0.25$) in order to reduce the computational load. The bottom two plots of Figure 2 show the evolution of the average of the error measure defined in equation 5.3 for the image and document data sets.

We observe the following:

- As for synthetic data sets, E-ANLS outperforms ANLS for all the values of the parameters.
- Since we have removed the values of the parameters performing worst, the gap between the best and worst variants of E-ANLS is reduced.
- For the image data sets, the variant with $hp = 1$ performs best, although the variants with $hp = 2, 3$ do not perform much worse.
- For the document data sets, the variants with $hp = 2, 3$ perform best (in terms of final error). This can be explained by the fact that NMF problems for sparse matrices are more difficult, as there are more local minima with rather different objective function values (see section 5.4.3 for more numerical experiments). Hence, the final error reports the algorithm that found the best solution in most of the 60 cases (6 data sets, 10 initializations per data set). In terms of speed of convergence, most E-ANLS variants behave similarly (converging within 80 iterations, while ANLS has not converged within 100 iterations).

In the final numerical experiments, we will use $\beta_0 = 0.5$, $\eta = 1.5$ and $(\gamma, \bar{\gamma}) = (1.1, 1.05)$ for E-ANLS. We will keep both variants $hp = 1, 3$.

5.3.2 Extrapolated A-HALS (E-A-HALS). The top two plots of Figure 3 show the evolution of the average of the error measure defined in equation 5.3 for the low-rank and full-rank synthetic data sets. For these experiments, we have also tested the value $(\gamma, \bar{\gamma}) = (1.005, 1.001)$ (as we will see, that smaller value of these parameters perform better). We observe the following:

- For the low-rank synthetic data, with $hp = 2, 3$ and well-chosen parameters for the update of β (e.g., $\beta_0 = 0.50$, $\eta = 1.5$, $\gamma = 1.01$), E-A-HALS performs much better than A-HALS. (Note, however, that it is not able to find solutions with error as small as E-ANLS within 500 iterations.)

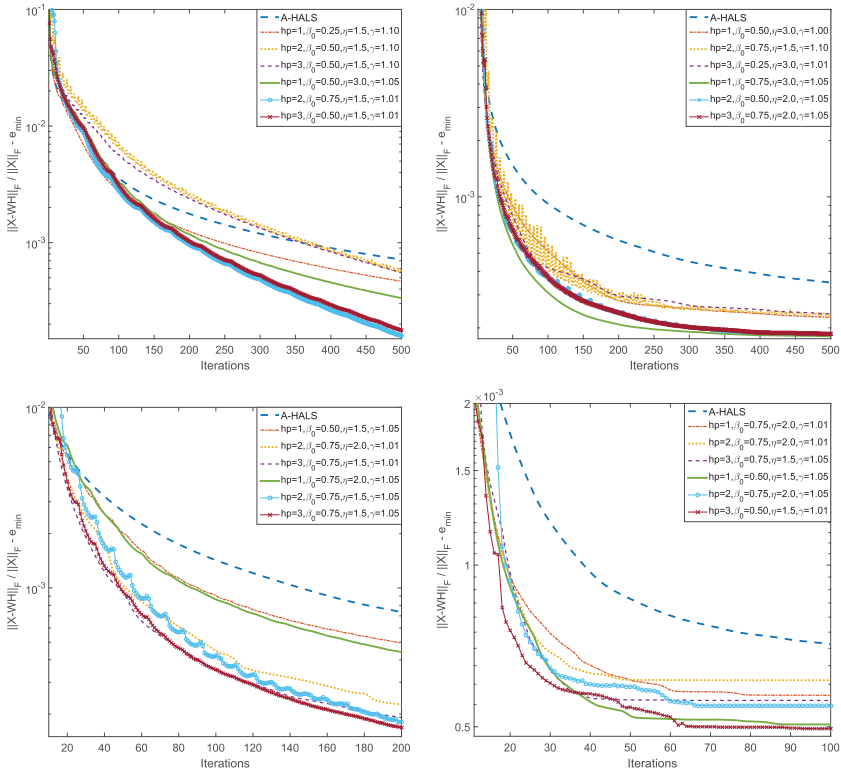

Figure 3: Extrapolation scheme applied with A-HALS on the low-rank (top left) and full-rank (top right) synthetic data sets and on the image (bottom left) and document (bottom right) data sets. For each value of hp, we display the corresponding best- and worst-performing variant.

- For the full-rank synthetic data, the variant with $hp = 1$ performs slightly better, although the final solutions of the three extrapolated variants have similar error.
- The best value for γ is either 1.01 or 1.05, smaller than for E-ANLS. This can be explained by the fact that A-HALS does not solve the NNLS subproblems exactly, and the extrapolation should not be as aggressive as for ANLS. As for E-ANLS, E-HALS is not too sensitive to the parameters η and β_0.

We now perform the same experiment on image and document data sets except with fewer parameters (we do not test $\gamma = 1.005, 1.1$, $\eta = 3$, $\beta_0 = 0.25$). The bottom two plots of Figure 3 show the evolution of the error

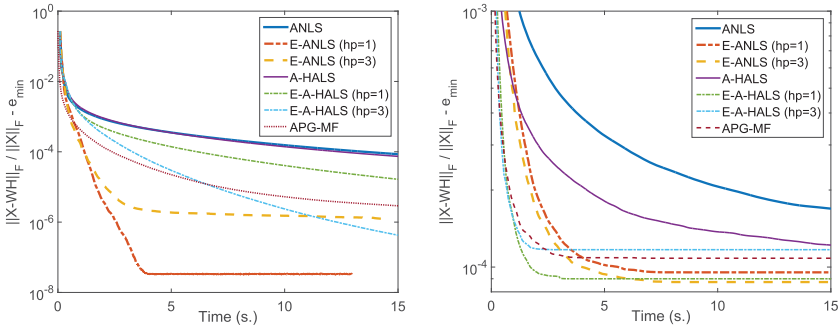

Figure 4: Average value of the error measure, equation 5.3, of ANLS, A-HALS, and their extrapolated variants applied on low-rank (left) and full-rank (right) synthetic data sets.

measure defined in equation 5.3 for the image and document data sets. We observe the following:

- For the image data sets, the variant $hp = 1$ performs worse than $hp = 2, 3$, which perform similarly (in terms of speed of convergence).
- For the document data sets, we observe similar behavior as for ANLS: all extrapolated variants converge much faster than HALS, but they converge to different solutions (being on average less than 0.1% away from the lowest relative error).

In the final numerical experiments, we use $\beta_0 = 0.5$, $\eta = 1.5$, and $(\gamma, \bar{\gamma}) = (1.01, 1.005)$ for E-A-HALS. We keep both variants $hp = 1, 3$.

5.4 Extensive Numerical Experiments and Comparison of E-ANLS and E-HALS. We now compare ANLS, A-HALS and their extrapolated variants on the same data sets. We also compare these algorithms with the extrapolated alternating projected gradient method for NMF proposed by Xu and Yin (2013) and referred to as APG-MF.

5.4.1 Synthetic Data Sets. Figure 4 displays the evolution of the average error for the low-rank and full-rank synthetic data sets, where the NMF algorithms were run for 15 seconds. Table 3 (resp. 4) reports the average error, standard deviation, and a ranking among the final solutions obtained by the different algorithms for the low-rank (resp. full-rank) synthetic data sets.

For low-rank synthetic data sets, these results confirm what we have observed previously: E-ANLS ($hp = 1$) is able to significantly accelerate ANLS and obtain solutions with very small error extremely fast (in less than 4 seconds). The acceleration of HALS is not as important, but it is significant.

Table 3: Comparison of the Final Relative Error Obtained by the NMF Algorithms on the Low-Rank Synthetic Data Sets: Average Error, Standard Deviation, and Rankings among the 100 Runs (100 Data Sets).

Algorithm	Mean ± SD	Ranking
ANLS	$5.612\,10^{-5} \pm 7.414\,10^{-5}$	(0, 0, 0, 3, 7, 40, 50)
E-ANLS ($hp = 1$)	$\mathbf{2.618\,10^{-8} \pm 3.657\,10^{-8}}$	(96, 4, 0, 0, 0, 0, 0)
E-ANLS ($hp = 3$)	$1.207\,10^{-6} \pm 1.162\,10^{-5}$	(67, 24, 7, 1, 1, 0, 0)
A-HALS	$4.547\,10^{-5} \pm 6.299\,10^{-5}$	(1, 0, 0, 4, 10, 41, 44)
E-A-HALS ($hp = 1$)	$7.825\,10^{-6} \pm 1.531\,10^{-5}$	(3, 0, 6, 31, 41, 13, 6)
E-A-HALS ($hp = 3$)	$1.181\,10^{-7} \pm 3.679\,10^{-7}$	(48, 8, 37, 7, 0, 0, 0)
APG-MF	$2.032\,10^{-6} \pm 5.770\,10^{-6}$	(0, 0, 3, 50, 41, 6, 0)

Notes: The ith entry of the vector indicates the number of times the algorithm generated the ith best solution. Observe that all algorithms are able to compute the best solution at least a few times; this happens when they compute an exact solution with $X = WH$. Numbers in bold indicate the best performance on average.

Table 4: Comparison of the Final Relative Error Obtained by the NMF Algorithms on the Full-Rank Synthetic Data Sets: Average Error, Standard Deviation, and Rankings among the 100 Runs (10 Data Sets, 10 Initializations Each).

Algorithm	Mean ± SD	Ranking
ANLS	$0.423858 \pm 1.183\,10^{-3}$	(4, 9, 9, 12, 22, 21, 23)
E-ANLS ($hp = 1$)	$0.423795 \pm 1.161\,10^{-3}$	(16, 18, 18, 9, 15, 10, 14)
E-ANLS ($hp = 3$)	$\mathbf{0.423787 \pm 1.158\,10^{-3}}$	(18, 11, 17, 21, 16, 9, 8)
A-HALS	$0.423815 \pm 1.171\,10^{-3}$	(18, 12, 11, 18, 13, 13, 15)
E-A-HALS ($hp = 1$)	$0.423790 \pm 1.162\,10^{-3}$	(17, 17, 16, 17, 15, 10, 8)
E-A-HALS ($hp = 3$)	$0.423817 \pm 1.184\,10^{-3}$	(12, 14, 16, 11, 11, 22, 14)
APG-MF	$0.423808 \pm 1.183\,10^{-3}$	(16, 18, 13, 12, 8, 15, 18)

Notes: The ith entry of the vector indicates the number of times the algorithm generated the ith best solution. The numbers in bold indicate the best performance on average.

E-ANLS ($hp = 1$) is able to obtain the best solutions in 96 of the 100 runs while always being among the two best, while ANLS and A-HALS are among the worst ones in most cases. APG-MF never generates the best or the second-best solution.

For full-rank synthetic data sets, we observe that all algorithms obtain a similar final relative error (see Table 4), all of them being on average around 0.01% away from the best solution, and there is no clear winner between the extrapolated variants. In fact, there is a priori no reason to believe that an algorithm will converge to a better solution in general as NMF is a difficult nonconvex optimization problem (Vavasis, 2010). In terms of speed of convergence, E-A-HALS variants converge the fastest (about 3 seconds),

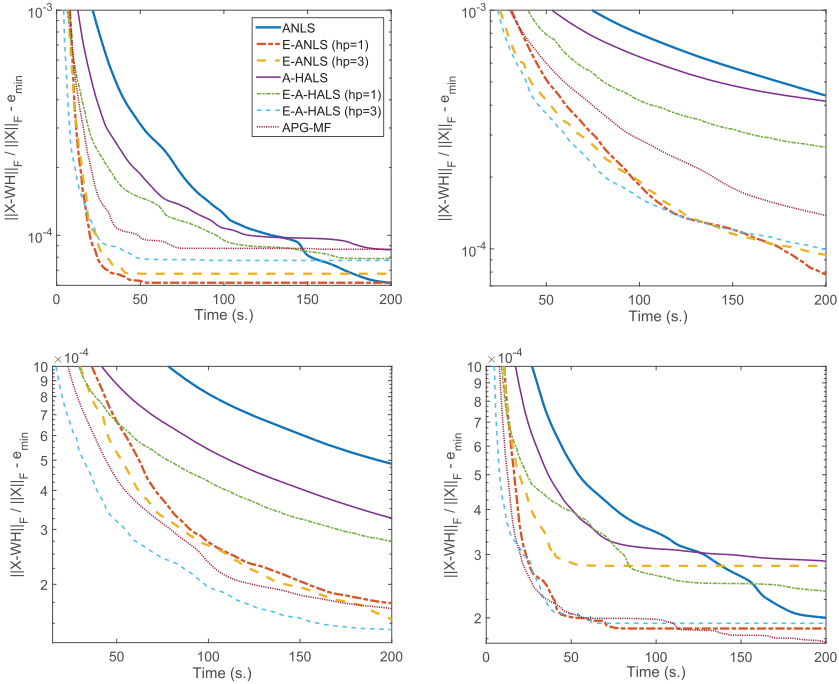

Figure 5: Average value of the error measure, equation 5.1, of ANLS, A-HALS, and their extrapolated variants applied on the four image data sets: CBCL (top left), Umist (top right), ORL (bottom left), Frey (bottom right).

followed by APG-MF (about 4 seconds) and the E-ANLS variants (about 8 seconds), while A-HALS and ANLS require more than 20 seconds.

5.4.2 Dense Image Data Sets. We now run the algorithms for 200 seconds on the four image data sets (see Figure 5, which displays the evolution of the average error measure, equation 5.1, for each data set, and Table 5, which compares the final errors obtained by the different algorithms.

We observe the following:

- E-A-HALS ($hp = 3$) has the fastest initial convergence speed, followed by E-ANLS variants and APG-MF. As in the preliminary numerical experiments, E-A-HALS ($hp = 1$) is able to accelerate A-HALS but not as much as E-A-HALS ($hp = 3$).
- In terms of final error, there is no clear winner between the extrapolated variants (similarly as for the full-rank synthetic data sets), while ANLS clearly performs the worst.

Table 5: Comparison of the Final Relative Error Obtained by the NMF Algorithms on the Image Data Sets: Average Error, Standard Deviation and Rankings among the 40 Runs (4 Data Sets, 10 Initializations Each).

Algorithm	Mean $\pm$ SD	Ranking
ANLS	$0.110703 \pm 2.964\,10^{-2}$	(3, 3, 3, 5, 5, 8, 13)
E-ANLS ($hp = 1$)	$\mathbf{0.110547 \pm 2.958\,10^{-2}}$	(9, 12, 7, 5, 4, 2, 1)
E-ANLS ($hp = 3$)	$0.110570 \pm 2.956\,10^{-2}$	(9, 6, 5, 7, 2, 8, 3)
A-HALS	$0.110690 \pm 2.956\,10^{-2}$	(1, 4, 4, 2, 3, 13, 13)
E-A-HALS ($hp = 1$)	$0.110634 \pm 2.958\,10^{-2}$	(4, 2, 2, 4, 17, 7, 4)
E-A-HALS ($hp = 3$)	$0.110552 \pm 2.956\,10^{-2}$	(5, 10, 11, 8, 3, 0, 3)
APG-MF	$0.110559 \pm 2.956\,10^{-2}$	(9, 3, 8, 9, 6, 2, 3)

Notes: The ith entry of the vector indicates the number of times the algorithm generated the ith best solution. The numbers in bold indicate the best performance on average.

To conclude, we see that the extrapolation scheme is particularly beneficial to ANLS, which is significantly accelerated, and even able to outperform E-A-HALS in some cases (while A-HALS performs in general much better than ANLS, as already pointed out in Gillis & Glineur, 2012). Although APG-MF outperforms ANLS (as already observed by (Xu & Yin, 2013)) and A-HALS, it is in general outperformed by the other extrapolated variants.

5.4.3 Sparse Document Data Sets. We now run the algorithms for 200 seconds on the six document data sets. Figure 6 displays the evolution of the average error measure, equation 5.1, for each data set, and Table 6 compares the final errors obtained by the different algorithms.

We observe the following:

- E-A-HALS variants have the fastest initial convergence speed converging on average in about 10 seconds, followed by A-HALS, which sometimes takes much more time to stabilize (e.g., more than 50 seconds for the classic data set). APG-MF does not converge as fast as E-A-HALS variants. E-ANLS variants converge much faster than ANLS but sometimes take more than 30 seconds to stabilize.
- In terms of final error, there is no clear winner, although A-HALS and E-A-HALS ($hp = 3$) most of the time give the best solution (15 our of 60 cases). APG-MF tends to generate the worst solutions (17 out of the 60 cases) and performs similarly as ANLS in this respect.

For sparse data sets, E-A-HALS is the best option for which both variants ($hp = 1, 3$) perform similarly. APG-MF and ANLS and its extrapolated variants are less effective in this case.

Remark 4 (choice of hp). At this point, we do not have a good theoretical understanding to justify the choice of hp. From the numerical experiments,

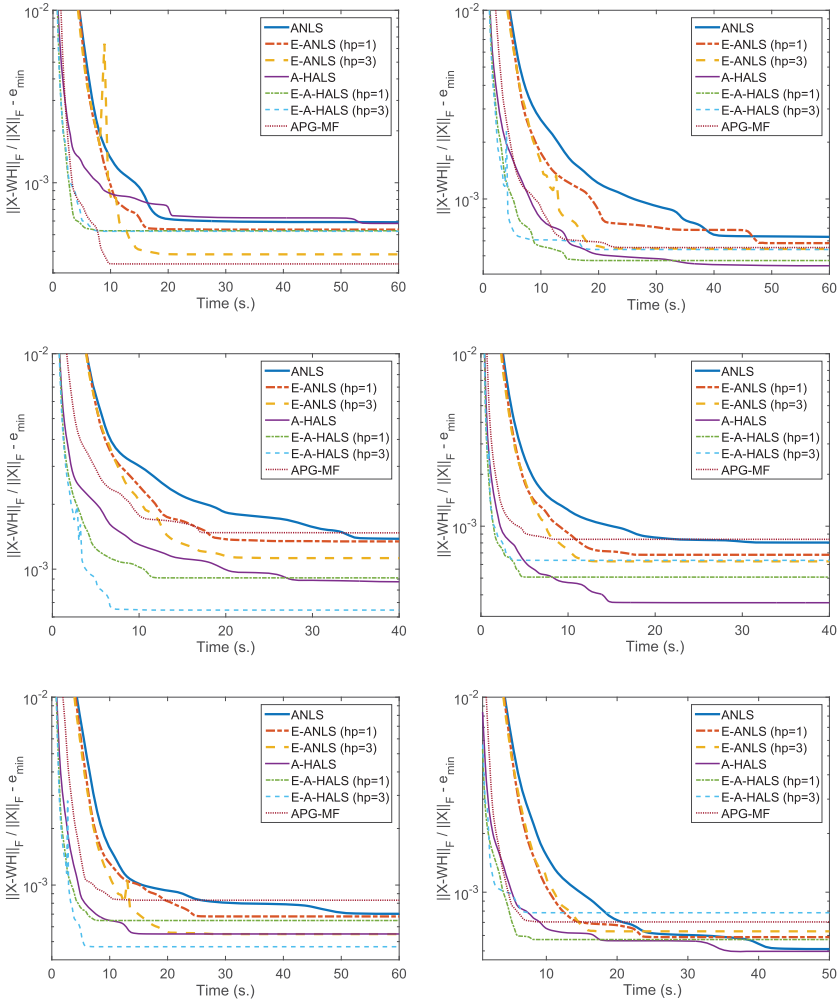

Figure 6: Average value of the error measure, equation 5.1, of ANLS, A-HALS, and their extrapolated variants applied on the six documents data sets: Classic (top left), Sports (top right), Reviews (middle left), Hitech (middle right), ohscal (bottom left), la1 (bottom right).

it is clear that $hp = 2$ should be avoided as it performs in most cases worse than $hp = 1, 3$ while being a more aggressive variant (no projection and extrapolation after the update of both W and H; see sections 5.1 and 5.2). Comparing $hp = 1$ and $hp = 3$, there is no clear winner, and performance varies from one experiment to another. Understanding this behavior and possibly

Table 6: Comparison of the Final Relative Error Obtained by the NMF Algorithms on the Document Data Sets: Average Error, Standard Deviation, and Rankings among the 60 Runs (6 Data Sets, 10 Initializations Each).

Algorithm	Mean ± SD	Ranking
ANLS	$0.850433 \pm 3.186\,10^{-2}$	(5, 3, 12, 6, 9, 11, 14)
E-ANLS ($hp = 1$)	$0.850417 \pm 3.187\,10^{-2}$	(7, 8, 6, 12, 13, 12, 2)
E-ANLS ($hp = 3$)	$0.850324 \pm 3.189\,10^{-2}$	(9, 11, 6, 9, 15, 6, 4)
A-HALS	$\mathbf{0.850232 \pm 3.198\,10^{-2}}$	(15, 11, 9, 8, 7, 7, 3)
E-A-HALS ($hp = 1$)	$0.850287 \pm 3.198\,10^{-2}$	(13, 13, 12, 6, 7, 6, 3)
E-A-HALS ($hp = 3$)	$0.850281 \pm 3.204\,10^{-2}$	(15, 11, 11, 4, 5, 9, 5)
APG-MF	$0.850471 \pm 3.183\,10^{-2}$	(5, 5, 9, 10, 5, 9, 17)

Note: The ith entry of the vector indicates the number of times the algorithm generated the ith best solution.

designing a better strategy (e.g., using a hybridization) is a topic for further research.

6 Conclusion

In this letter, we have proposed an extrapolation scheme for NMF algorithms to significantly accelerate their convergence. We have focused on two state-of-the-art NMF algorithms: ANLS (Kim & Park, 2011) and A-HALS (Gillis & Glineur, 2012). The main conclusions are the following:

- In all cases, the extrapolated variants significantly outperform the original algorithms.
- For randomly generated low-rank matrices, E-ANLS, the extrapolated variant of ANLS, allows a significant acceleration; it is able to compute solutions with very small relative errors ($\approx 10^{-8}$) in all cases, while the other approaches fail to do so.
- For dense data sets, E-ANLS and E-A-HALS perform similarly, although A-HALS performs much better than ANLS. This is interesting: the extrapolated variants allowed ANLS to get back on A-HALS.
- For sparse data sets, E-A-HALS performs the best and should be preferred to the other variants.
- The extrapolated projected gradient method proposed by Xu and Yin (2013) and referred to as APG-MF performs well but does not perform as well as the extrapolated variants proposed in this letter.

This work was mostly experimental. It would be crucial to understand the extrapolation scheme better from a theoretical point of view. In particular, can we prove convergence to a stationary point as done in Xu and Yin (2013)? And can we quantify precisely the acceleration as it has been done in the convex case? Further work also includes the use of extrapolation in

other settings, such as NMF with other objective functions such as the β divergences (Févotte & Idier, 2011), nonnegative tensor factorization (NTF; Cichocki et al., 2009) and symmetric NMF (Vandaele, Gillis, Lei, Zhong, & Dhillon, 2016).

Acknowledgments

We thank the anonymous reviewer for his or her insightful comments, which helped improve this letter. We were supported by the European Research Council (ERC starting grant 679515). N.G. was supported by the Fonds de la Recherche Scientifique and the Fonds Wetenschappelijk Onderzoek–Vlaanderen under EOS Project O005318F-RG47.

References

Beck, A., & Tetruashvili, L. (2013). On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, *23*(4), 2037–2060.

Chambolle, A., & Pock, T. (2015). A remark on accelerated block coordinate descent for computing the proximity operators of a sum of convex functions. *SMAI Journal of Computational Mathematics*, *1*, 29–54.

Chow, Y. T., Wu, T., & Yin, W. (2017). Cyclic coordinate-update algorithms for fixed-point problems: Analysis and applications. *SIAM Journal on Scientific Computing*, *39*(4), A1280–A1300.

Cichocki, A., & Phan, A.-H. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics*, *92*(3), 708–721.

Cichocki, A., Zdunek, R., & Amari, S. (2007). Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization. In M. E. Davies, C. J. James, S. A. Abdallah, & M. D. Plumbley (Eds.), *Lecture Notes in Computer Science*, Vol. 4666. *Independent component analysis and signal separation* (pp. 169–176). Berlin: Springer-Verlag.

Cichocki, A., Zdunek, R., Phan, A. H., & Amari, S.-i. (2009). *Nonnegative matrix and tensor factorizations: Applications to exploratory multi-way data analysis and blind source separation*. Hoboken, NJ: Wiley.

Erichson, N. B., Mendible, A., Wihlborn, S., & Kutz, J. N. (2018). Randomized nonnegative matrix factorization. *Pattern Recognition Letters*, *104* 1–7. doi:10.1016/j.patrec.2018.01.007

Fercoq, O., & Richtárik, P. (2015). Accelerated, parallel, and proximal coordinate descent. *SIAM Journal on Optimization*, *25*(4), 1997–2023.

Févotte, C., & Idier, J. (2011). Algorithms for nonnegative matrix factorization with the β-divergence. *Neural Computation*, *23*(9), 2421–2456.

Fu, X., Huang, K., Sidiropoulos, N. D., & Ma, W.-K. (2018). *Nonnegative matrix factorization for signal and data analytics: Identifiability, algorithms, and applications*. arXiv:1803.01257.

Fu, X., Huang, K., Yang, B., Ma, W. K., & Sidiropoulos, N. D. (2016). Robust volume minimization-based matrix factorization for remote sensing and document clustering. *IEEE Transactions on Signal Processing*, *64*(23), 6254–6268.

Gillis, N. (2014). The why and how of nonnegative matrix factorization. In J. Suykens, M. Signoretto, & A. Argyriou (Eds.), *Regularization, optimization, kernels, and support vector machines* (pp. 257–291). London: Chapman & Hall/CRC.

Gillis, N. (2017). Introduction to nonnegative matrix factorization. *SIAG/OPT Views and News*, *25*(1), 7–16.

Gillis, N., & Glineur, F. (2012). Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization. *Neural Computation*, *24*(4), 1085–1105.

Guan, N., Tao, D., Luo, Z., & Yuan, B. (2012). NeNMF: An optimal gradient method for nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, *60*(6), 2882–2898.

Hong, M., Wang, X., Razaviyayn, M., & Luo, Z.-Q. (2017). Iteration complexity analysis of block coordinate descent methods. *Mathematical Programming*, *163*(1–2), 85–114.

Hsieh, C.-J., & Dhillon, I. S. (2011). Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1064–1072). New York: ACM.

Kim, J., He, Y., & Park, H. (2014). Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework. *Journal of Global Optimization*, *58*(2), 285–319.

Kim, H., & Park, H. (2008). 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.

Kim, J., & Park, H. (2011). Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, *33*(6), 3261–3281.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, *401*(6755), 788.

Lin, C.-J. (2007). Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, *19*(10), 2756–2779.

Luenberger, D. G., & Ye, Y. (2015). *Linear and nonlinear programming*, 4th ed. Springer. https://web.stanford.edu/class/msande310/310trialtext.pdf.

Nesterov, Y. (2013). *Introductory lectures on convex optimization: A basic course*. New York: Springer Science & Business Media.

Odonoghue, B., & Candès, E. (2015). Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, *15*(3), 715–732.

O'Neill, M., & Wright, S. J. (2017). *Behavior of accelerated gradient methods near critical points of nonconvex problems*. arXiv:1706.07993.

Paquette, C., Lin, H., Drusvyatskiy, D., Mairal, J., & Harchaoui, Z. (2018). Catalyst for gradient-based nonconvex optimization. In A. Storkey, & F. Perez-Cruz (Eds.), *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*. Vol. 84: *Proceedings of Machine Learning Research*. http://proceedings.mit.press/v84

Vandaele, A., Gillis, N., Lei, Q., Zhong, K., & Dhillon, I. (2016). Efficient and non-convex coordinate descent for symmetric nonnegative matrix factorization. *IEEE Transactions on Signal Processing*, *64*(21), 5571–5584.

Vavasis, S. A. (2010). On the complexity of nonnegative matrix factorization. *SIAM Journal on Optimization*, *20*(3), 1364–1377.

Xu, Y., & Yin, W. (2013). 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.

Zhong, S., & Ghosh, J. (2005). Generative model-based document clustering: A comparative study. *Knowledge and Information Systems*, *8*(3), 374–384.

Received May 18, 2018; accepted September 26, 2018.