ACCELERATED ALGORITHMS FOR NONLINEAR MATRIX DECOMPOSITION WITH THE RELU FUNCTION

Giovanni Seraghiti†, *Atharva Awari*‡, *Arnaud Vandaele*‡, *Margherita Porcelli*†, *Nicolas Gillis*‡

† University of Bologna, Piazza di Porta San Donato 5, 40126 Bologna, Italy
‡ University of Mons, Rue de Houdain 9, 7000 Mons, Belgium

ABSTRACT

In this paper, we study the following nonlinear matrix decomposition (NMD) problem: given a sparse nonnegative matrix X, find a low-rank matrix Θ such that $X \approx f(\Theta)$, where f is an element-wise nonlinear function. We focus on the case where $f(\cdot) = \max(0, \cdot)$, the rectified linear unit (ReLU) nonlinear activation. We refer to the corresponding problem as ReLU-NMD. We first provide a brief overview of the existing approaches that were developed to tackle ReLU-NMD. Then we introduce two new algorithms: (1) aggressive accelerated NMD (A-NMD) which uses an adaptive Nesterov extrapolation to accelerate an existing algorithm, and (2) three-block NMD (3B-NMD) which parametrizes $\Theta = WH$ and leads to a significant reduction in the computational cost. We also propose an effective initialization strategy based on the nuclear norm as a proxy for the rank function. We illustrate the effectiveness of the proposed algorithms on synthetic and real-world data sets.

Index Terms— low-rank matrix factorization, nonlinearity, extrapolation, alternating minimization, nuclear norm.

1. INTRODUCTION

Low-rank matrix approximations are widely used in many fields such as data analysis and machine learning. When dealing with a large amount of data, stored in a matrix X, performing dimensionality reduction by approximating X by a low-rank matrix is frequent in applications such as data compression, interpretation, and visualization. In general, one wants to find a low-rank matrix Θ such that $X \approx \Theta$. Some well-known examples of this type of approximation are the truncated singular value decomposition (TSVD) [1], and nonnegative matrix factorization (NMF) [2]. In this context, there has been a recent emergence of nonlinear matrix decomposition (NMD), via its close connection with neural networks [3]. NMD looks for a low-rank matrix Θ such that $X \approx f(\Theta)$, where f is an elementwise nonlinear function; of particular interest is the case where f is the ReLU function, that is, $f(\cdot) = \max(0, \cdot)$, which is often used as an activation function in hidden layers of neural networks.

NG acknowledges the support by the F.R.S.-FNRS and the FWO (EOS no O005318F-RG47), by the ERC (grant no 101085607), by the the F.R.S.-FNRS under the PDR project T.0097.22, and by the Francqui Foundation.

The NMD problem we consider in this paper is the following: Given $X \in \mathbb{R}^{m \times n}$ and $r < \min(m, n)$, solve

$$\min_{\Theta \in \mathbb{R}^{m \times n}} \|X - \max(0, \Theta)\|_F^2 \quad \text{such that} \quad \operatorname{rank}(\Theta) = r. \quad (1)$$

We will refer to this problem as ReLU-NMD. ReLU-NMD makes sense only if X is nonnegative, since $\max(0, \Theta) \geq 0$. Moreover, X should be relatively sparse for ReLU-NMD to provide advantages compares to the TSVD: if X has mostly positive entries, the solution of ReLU-NMD will be similar to that of the TSVD, since Θ will need to contain mostly positive entries. For example, Saul showed that the identity matrix of any dimension can be exactly recovered with a rank-3 ReLU-NMD [3]. The reason is that $\max(0, \Theta)$ can be full rank although Θ has low rank; see Section 5 for other examples.

The objective function in (1) is neither differentiable nor convex, and the nonlinearity arising from the ReLU function makes the problem difficult to solve using a direct approach. ReLU-NMD has been recently investigated by Saul [3], where he introduced another formulation for ReLU-NMD as the following latent variable model:

$$\min_{Z, \Theta} \|Z - \Theta\|_F^2 \quad \text{such that} \quad \begin{cases} \operatorname{rank}(\Theta) = r, \\ \max(0, Z) = X. \end{cases} \quad (2)$$

The main advantage of this new formulation is that the additional latent variable Z allows one to move the nonlinearity from the objective function to the constraints, opening the possibility of exploring new solution strategies.

In [3], Saul presents two algorithms to solve (2): a naive algorithm, and an Expectation-Maximization (EM) algorithm. In a nutshell, they are both alternating minimization algorithms with respect to the two variables Z and Θ, and rely at each iteration on the computation of a rank-r TSVD of an m-by-n matrix; see Section 2 for more details. Furthermore, in [4], Saul uses a momentum acceleration step with fixed extrapolation parameter to accelerate the convergence of the algorithms.

Outline and contribution In this paper, we provide new effective algorithms for ReLU-NMD. In Section 2, we summarize the previous works on this problem, mostly that of Saul. In Section 3, we first include an adaptive momentum parameter estimation in the naive algorithm of Saul with the aim of optimally self-tuning the parameter along the iterations. Then we introduce the three-block NMD (3B-NMD) algorithm, which exploits the parametrization $\Theta = WH$, and then uses an accelerated block coordinate method to update W, H and Z. The main advantages of 3B-NMD is that it avoids the computation of a rank-r TSVD at each iteration by solving instead least squares problems, reducing the computational cost from $O(mnr^2)$ to $O(mnr)$ operations. In Section 4, we explain how the nuclear norm can be used to provide a good initialization to ReLU-NMD. Finally, we illustrate on synthetic and real-world data sets the effectiveness of the proposed algorithms compared to the state of the art in Section 5.

2. PREVIOUS ALGORITHMS BY SAUL

In this section, we briefly recall the two main algorithms proposed by Saul; the first one will be the starting point of our newly proposed algorithms in Section 3.

Naive algorithm and extrapolation A simple algorithm to tackle the reformulation (2) is alternating optimization [3]. Let $I_+ = \{(i,j) \mid X_{ij} > 0\}$ and $I_0 = \{(i,j) \mid X_{ij} = 0\}$. At each iteration, Z and Θ are computed alternatively: the optimal solution for Z, when Θ is fixed, is given by

$$Z_{ij} = \begin{cases} X_{ij} & \text{if} \quad (i,j) \in I_+, \\ \min(0, \Theta_{ij}) & \text{if} \quad (i,j) \in I_0. \end{cases} \tag{3}$$

The optimal solution for Θ when Z is fixed is the rank-r TSVD of Z. We will refer to this algorithm as Naive.

In [4], Saul accelerates the above simple scheme by an additional momentum term on the update of Z with fixed momentum parameter [5]. Denoting by Z^{k+1}, the k+1-th iterate, after Z^{k+1} is computed, it is updated as

$$Z^{k+1} \leftarrow Z^{k+1} + \alpha(Z^k - Z^{k-1}), \tag{4}$$

where $\alpha \in (0,1)$ and it the experiments we set $\alpha = 0.7$. We will refer to the accelerated Naive algorithm as A-Naive.

Expectation-Maximization (EM-NMD) In [3], Saul proposes a second more sophisticated EM algorithm, where the matrix Θ parameterizes the following Gaussian latent variable model for the data X: for all i,j, $\tilde{Z}_{ij} \sim \mathcal{N}(\Theta_{ij}, \sigma^2)$. It is assumed that the observation, Z, is a sample of $\tilde{Z}$, and the matrix X is obtained from the elementwise nonlinear mapping of Z: $X = \max(0, Z)$. The model is then estimated by maximizing the likelihood of the observation X in terms of the matrix Θ and variance σ^2. The overall log-likelihood under this model is given by

$$\log P(X|\Theta, \sigma^2) = \Sigma_{ij} \log P(X_{ij}|\Theta_{ij}, \sigma^2). \tag{5}$$

The parameters Θ and σ^2 are estimated by maximizing this sum. To do this, EM is used: it alternates between two steps: the E-step computes the posterior means and variances of the model latent variables, and the M-step uses these posterior statistics to re-estimate the model parameters. These steps are rather technical, and we refer the interested reader to [3] for more details. Note that the M-step requires the computation of a rank-r TSVD, as in the naive approach, and hence requires $O(mnr^2)$ operations. In [4] a momentum step as in (4) is added to the algorithm, we will refer to it as A-EM.

3. TWO NEW ALGORITHMS FOR RELU-NMD

Let us present our two new algorithms for ReLU-NMD.

3.1. Aggressive momentum algorithm (A-NMD)

In the naive algorithm, Saul used a Polyak-type extrapolation with a fixed momentum parameter and only extrapolated the variable Z; see Section 2. Here we adopt a more aggressive Nesterov-type extrapolation with a heuristic approach to tune this parameter, as we set

$$Z^{k+1} \leftarrow Z^{k+1} + \beta_k(Z^{k+1} - Z^k), \tag{6}$$

where β_k is chosen adaptively. Also note that we use extrapolation for both variables, Z and Θ. The adaptive choice of the momentum parameter allows the algorithm to be less sensitive to that parameter, and adapt depending on the problem at hand. To do so, we follow the scheme in [6] where it was used for NMF. In that scheme, the momentum parameter, β_k at iteration k, is updated based on the decrease/increase of the objective function as follows. Fix the hyperparameters $1 < \bar{\gamma} < \gamma < \eta$. The momentum parameter is multiplied by γ as long as the objective function is decreasing, unless it reaches the adaptive upper bound $\bar{\beta}$. If the objective function decreases, we set $\bar{\beta} = \min(1, \bar{\gamma}\bar{\beta})$, meaning that we increase $\bar{\beta}$ by a factor $\bar{\gamma} < \gamma$. On the contrary, if at iteration k the error increases, the momentum parameter is divided by the factor η and we update the upper bound $\bar{\beta}$ as β_{k-1}, meaning that $\bar{\beta}$ keeps track of the latest value of β that allowed the decrease of the objective function. Algorithm 1 summarizes this strategy which we refer to as A-NMD. In the numerical experiments, we will use the parameters $\bar{\gamma} = 1.05 < \gamma = 1.1 < \eta = 2.5$.

3.2. Three-block NMD algorithm (3B-NMD)

All the previously described algorithms require the computation of a rank-r TSVD at each step. To avoid this relatively expensive step, we substitute $\Theta \in \mathbb{R}^{m \times n}$ by the product WH,

Algorithm 1 Aggressive momentum NMD (A-NMD)

Input: X, Z^0, Θ^0, r, $1 < \bar{\gamma} < \gamma < \eta$, $\beta_0 \in (0,1)$, maxit.
Output: A rank-r matrix Θ s.t. $X \approx \max(0,\Theta)$.

```
 1: Set $\bar{\beta} = 1$, $Z^k_{ij} = X_{ij}$ for $(i,j) \in I_+$ and for $k = 0,1$.
 2: for $k = 0,1,\ldots,$ maxit do
 3:     $Z^{k+1}_{ij} = \min(0, \Theta^k_{ij})$ for $(i,j) \in I_0$.
 4:     $Z^{k+1} \leftarrow Z^{k+1} + \beta_k(Z^{k+1} - Z^k)$.
 5:     $[U,D,V] = \text{TSVD}(Z^{k+1}, r)$.
 6:     $\Theta^{k+1} = UDV^T$.
 7:     $\Theta^{k+1} \leftarrow \Theta^{k+1} + \beta_k(\Theta^{k+1} - \Theta^k)$.
 8:     if $\|X - \max(0,\Theta^{k+1})\|_F < \|X - \max(0,\Theta^k)\|_F$ then
 9:         $\beta_{k+1} = \min(\bar{\beta}, \gamma\beta_k)$, $\bar{\beta} = \min(1, \bar{\gamma}\bar{\beta})$.
10:     else
11:         $\beta_{k+1} = \beta_k \backslash \eta$, $\bar{\beta} = \beta_{k-1}$,
12:         $Z^{k+1} = Z^k$, $\Theta^{k+1} = \Theta^k$.
13:     end if
14: end for
15: $\Theta = \Theta^{k+1}$.
```

where $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r\times n}$. Hence we reformulate (2) as follows

$$\min_{Z,W,H} \|Z - WH\|_F^2 \quad \text{such that} \quad \max(0,Z) = X.$$

As for Θ, the minimization subproblems for W and H have closed-form solutions; in fact, they can be obtained simply by solving matrix least square problems which require $O(mnr)$ operations, instead of the $O(mnr^2)$ operations for the TSVD. To accelerate this block-coordinate descent method scheme, we also use extrapolation, after the computation of Z and of $\Theta = WH$; see Algorithm 2 (3B-NMD).

Algorithm 2 Momentum three-block NDM (3B-NMD)

Input: X, Z^0, W^0, H^0,r, β, maxit.
Output: Two matrices W and H s.t. $X \approx \max(0, WH)$.

```
 1: Set $Z^k_{ij} = X_{ij}$ for $(i,j) \in I_+$ and $k = 0,1$.
 2: for $k = 0,1,\ldots,$ maxit do
 3:     $Z^{k+1}_{ij} = \min(0, \Theta^k_{ij})$ for $(i,j) \in I_0$.
 4:     $Z^{k+1} \leftarrow Z^{k+1} + \beta(Z^{k+1} - Z^k)$.
 5:     $W^{k+1} \leftarrow \text{argmin}_W \|Z^{k+1} - WH^k\|_F^2$.
 6:     $H^{k+1} \leftarrow \text{argmin}_H \|Z^{k+1} - W^{k+1}H\|_F^2$.
 7:     $\Theta^{k+1} \leftarrow W^{k+1}H^{k+1}$
 8:     $\Theta^{k+1} \leftarrow \Theta^{k+1} + \beta(\Theta^{k+1} - \Theta^k)$.
 9: end for
10: $W = W^{k+1}$, $H = H^{k+1}$.
```

Note that 3B-NMD does not use an adaptive strategy for the momentum parameter β, because we have observed it is not as effective as in the naive case described in the previous section. This is a topic for further research. In the numerical experiments, we will use $\beta = 0.7$ which performs well in practice.

4. INITIALIZATION WITH THE NUCLEAR NORM

In this section, we provide an initialization strategy for Θ using the nuclear norm. The nuclear norm of a matrix X is the sum of its singular values, and is denoted $\|X\|_* = \sum_i \sigma_i(X)$. The nuclear norm has been used as a convex surrogate of the rank function, akin to the ℓ_1 norm used as a convex surrogate for the ℓ_0 norm; see [7] and the references therein.

Assuming there exists an exact rank-r ReLU-NMD but we do not know the rank r, the rank identification problem can be reformulated as follows:

$$\min_{\Theta} \text{rank}(\Theta) \ \text{ such that } \ X = \max(0,\Theta), \tag{7}$$

which is a hard problem in general. Replacing the rank with the nuclear norm, we obtain the following convex relaxation:

$$\begin{aligned}\min_{\Theta} \|\Theta\|_* \ \text{ such that } \ \Theta_{ij} &= X_{ij} \text{ for } (i,j) \in I_+, \\ \Theta_{ij} &\leq 0 \text{ for } (i,j) \in I_0.\end{aligned} \tag{8}$$

We are looking for Θ that matches the positive entries of X while having the smallest possible nuclear norm, and hence, hopefully, a small rank. To solve (8), we resort to a standard projected subgradient strategy [7], updating Θ as follows

$$\Theta_{k+1} = \Pi(\Theta_k - \alpha_k Y_k), \qquad Y_k \in \partial\|\Theta_k\|_*,$$

where $\Pi(\cdot)$ is the projection onto the feasible set (which is easy to compute), and $\partial\|\Theta_k\|_*$ is a subgradient of the nuclear norm at Θ_k, given by [8]

$$\partial\|\Theta\|_* = \Big\{UV^T + P : P \text{ and } \Theta \text{ have orthogonal row and column spaces, and } \|P\| \leq 1\Big\}, \tag{9}$$

where $(U,\Sigma,V) \in \mathbb{R}^{m\times r} \times \mathbb{R}^{r\times r} \times \mathbb{R}^{n\times r}$ is a TSVD of Θ and $\|.\|$ is the operator norm (or induced 2-norm) of a matrix. In our implementation, we used $P = 0$.

There are several possibilities for the choice of the stepsizes α_k. A standard choice that guarantees convergence is to use a diminishing stepsize. However, we prefer to use a more aggressive backtracking approach, since we only needed to perform a few iterations of the subgradient algorithm as it was sufficient to obtain a significant decrease in the nuclear norm. Also, the solution is only used for initialization and hence we do not need high accuracy. Moreover, the solution of (8) is not guaranteed to be of rank smaller than r, and hence we use the rank-r TSVD of the last iterate as an initialization for the ReLU-NMD algorithms presented in the previous sections. Note that it would be however rather interesting to explore conditions under which the solution of (8) recovers that of (7), similarly as done in the affine rank minimization literature [7].

5. NUMERICAL EXPERIMENTS

We compare the following algorithms for ReLU-NMD: A-NMD (Algorithm 1), 3B-NMD (Algorithm 2), Naive-NMD, A-Naive-NMD, EM-NMD and A-EM by[1] Saul [3] (see Section 2). As a baseline, we will also report the result of the projection of the TSVD, that is, $\max(0, X_r)$ where X_r is the rank-r TSVD of X. All tests are preformed using Matlab R2021b on a laptop Intel CORE i5-1135G7 @ 2.40GHz 8GB RAM. In the following, we perform experiments on synthetic data sets, and the MNIST and CBCL data sets. The code is available online from `https://gitlab.com/ngillis/ReLU-NMD`.

5.1. Synthetic data

The matrix $X \in \mathbb{R}^{m \times n}$ is generated as $X = \max(0, WH)$, where the entries of $W \in \mathbb{R}^{m\times r}$ and $H \in \mathbb{R}^{r \times n}$ are generated from the normal distribution, that is, `W = randn(m,r)` and `H = randn(r,n)` in MATLAB. Since the probability for X to have a positive entry is equal to that of having a negative entry (by symmetry), X has, on average, 50% of its entries equal to zero. All algorithms are stopped when

$$\text{relative error} = \frac{\|X - \max(0,\Theta)\|_F}{\|X\|_F} \leq 10^{-4}, \qquad (10)$$

where Θ is the current solution. Note that, given the non-convexity of ReLU-NMD, there is no guarantee that an algorithm converges to such a small relative error. However, for the synthetic data as generated above, this is always the case. It would be interesting to explain this behavior, e.g., maybe there are no spurious local minima with high probability, as for other non-convex problems [9].

Initialization Let us first validate the effectiveness of the nuclear norm initialization. To do so, we compare:
• Random initialization where Θ is a rank-r matrix, generated in the same way as X, and which we scale optimally as follows $\Theta \leftarrow \alpha^* \Theta$ where

$$\alpha^* = \text{argmin}_\alpha \|X - \alpha \max(0,\Theta)\|_F = \frac{\langle X, \max(0,\Theta)\rangle}{\|\max(0,\Theta)\|_F^2}.$$

• The initialization Θ taken as the rank-r TSVD of X.
• The nuclear norm minimization strategy; see Section 4. We perform 3 iterations of the projected subgradient method (which we observe is a good tradeoff between computational cost and reduction in the error). We initialize the nuclear norm approach using the random initialization and optimal scaling as described above.

Table 1 reports the relative errors of all three initializations for two values of r, with $m = n = 500, 1000, 1500, 2000$. We observe that the nuclear norm allows an initial solution

[1]We thank Lawrence Saul for providing us with his MATLAB codes.

	$r=8$			$r=16$		
$m=n$	rand	TSVD	$\|.\|_*$	rand	TSVD	$\|.\|_*$
500	0.95	0.40	**0.36**	0.95	0.36	**0.32**
1000	0.95	0.41	**0.38**	0.95	0.37	**0.33**
1500	0.95	0.41	**0.38**	0.95	0.38	**0.33**
2000	0.95	0.41	**0.38**	0.95	0.38	**0.33**

Table 1. Initial relative error of three initializations: random initialization and scaling, the rank-r TSVD, and the nuclear norm initialization described in Section 4 ($\|.\|_*$).

with significantly smaller relative error: a reduction from a factor 2 to 3 compared to a random initialization, and about 10% improvement compared to the TSVD. Hence, in the remaining of the paper, we will use the nuclear norm initialization for all algorithms in all experiments.

Effectiveness of A-NMD Let us first show the effectiveness of the adaptive strategy used in A-NMD to accelerate the naive scheme by Saul described in Section 2. Table 2 reports the total time and iterations needed to reach a relative error as in (10). We considered averaged values over 5 different synthetic matrices, with 5 different random initializations, post-processed using the nuclear norm algorithm.

	Naive		A-Naive		A-NMD	
Size	time	iter	time	iter	time	iter
500	1.84	110	0.78	44	**0.57**	32
1000	7.21	87	2.85	33	**2.28**	27
1500	11.4	80	4.40	29	**3.71**	25
2000	21.2	78	8.19	28	**6.80**	24

Table 2. Average computational time needed to satisfy condition in (10) on synthetic data with $r = 32$.

We observe that A-NMD outperforms Naive and A-Naive: it about 4 times faster than Naive, and 20% faster, on average, than A-Naive. In the following experiments, we will therefore only compare the other algorithms with A-NMD.

Effectiveness of 3B-NMD Table 3 reports the total time and iterations needed to satisfy condition in (10), for a fixed rank $r = 32$, for the EM algorithm of Saul (EM-NMD), its accelerated variant (A-EM), as well as our proposed algorithms A-NMD and 3B-NMD. Table 4 reports the same quantities for fixed dimensions $m = n = 1000$, but with different values of the rank, r. We observe that A-EM performs better than EM-NMD, as expected, and hence we will report only the results for A-EM in the section on real-world data sets. Then, we observe that A-NMD performs better than A-EM (almost twice faster in all cases), while 3B-NMD outperforms all other algorithms, being more than 10 times faster than A-EM.

	A-NMD		3B-NMD		EM-NMD		A-EM	
Size	time	iter	time	iter	time	iter	time	iter
500	0.64	33	**0.08**	23	2.2	101	1.0	43
1000	2.4	27	**0.24**	24	8.1	78	3.9	36
1500	3.6	24	**0.46**	24	12.8	70	6.5	35
2000	6.7	25	**0.81**	24	22.1	66	11.6	34

Table 3. Time needed to satisfy condition in (10) for synthetic matrices of increasing dimension with $r = 32$.

	A-NMD		3B-NMD		EM-NMD		A-EM	
r	time	iter	time	iter	time	iter	time	iter
8	2.0	33	**0.22**	22	10.1	97	4.4	42
16	2.0	24	**0.20**	24	7.7	77	3.5	36
32	2.2	23	**0.22**	24	7.4	72	3.6	33
64	2.5	23	**0.25**	25	8.4	66	3.7	32

Table 4. Computational time needed to satisfy condition in (10) when approximating synthetic data of fixed size $n = m = 1000$ for different values of the rank, r.

5.2. MNIST data set

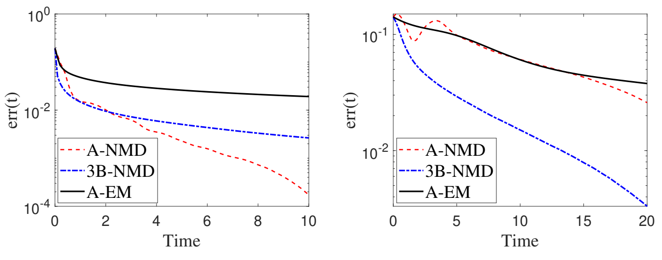
The experiments in this section are computed on 28×28 grayscale images of MNIST handwrittendigits [10], where X is generated by concatenating vectorized images into a matrix of size $784 \times n$ where $n = 500$ or $n = 50000$. To compare the ReLU-NMD algorithms, we will use the following quantity

$$err(t) = \frac{\|X - \max(0, \Theta(t))\|_F}{\|X\|_F} - e_{min}, \qquad (11)$$

where $\Theta(t)$ the solution at time t, and e_{min} is the smallest relative error obtained by all algorithms within the allotted time. Since $err(t)$ converges to zero for the algorithm that computed the best solution, we can represent the error in log scale. Figure 1 (a) displays the results on the MNIST data set with $r = 32$ with 500 images (50 images of each digit), with a timelimit of 10 seconds. Although 3B-NMD converges initially faster, A-NMD eventually catches up and generates the best solution. As before, A-EM is outperformed.

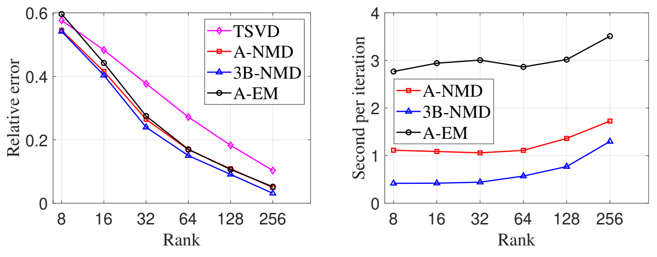
Figure 1 (b) displays the results on the MNIST data set with $r = 32$ with all the 50000 images, with a timelimit of 20 seconds. In this case, 3B-NMD converges initially faster and A-NMD does not have time to catch up. In any case, 3B-NMD and A-NMD both perform well, outperforming the state of the art algorithm A-EM.

Figure 2 compares the algorithms with the baseline, TSVD, in terms of relative error as the rank increases, and provides the average time per iteration, for 50000 images of MNIST and a runtime of 20 seconds. These results confirm the effectiveness of the 3B-NMD in order to deal with large data, the cost per iteration being smaller than that of A-NMD and A-EM. In addition, note that all the ReLU-NMD algorithms approximate the dataset with considerably higher accuracy than the TSVD, as expected.



(a) $n = 500$. (b) $n = 50000$.

Fig. 1. Average value of the error (11) of A-NMD, 3B-NMD and EM-NMD on small (a) and large (b) portion of the MNIST data set.



(a) Relative error. (b) Time (seconds).

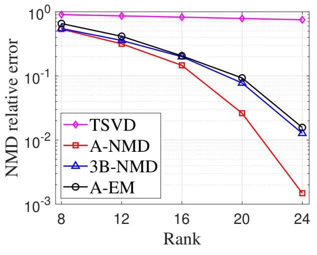
Fig. 2. Final relative error on $m = 50000$ images from MNIST dataset, after 20 seconds and average iteration time for increasing value of the rank r.

5.3. Compression of sparse NMF basis

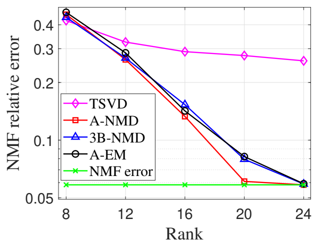
Another application of ReLU-NMD, which is new to the best of our knowledge, is the compression of sparse nonnegative dictionaries, e.g., the factors generated by NMF. Let us illustrate this on the CBCL data set, used in the seminal paper of Lee and Seung [11]. Each column of the data matrix $X \in \mathbb{R}^{361 \times 2429}$ contains a vectorized facial image of size 19×19. The NMF decomposition, $X \approx UV$ where $U \geq 0$ and $V \geq 0$, allows one to extract sparse facial features as the columns of U. To do so, we have used the regularized minimum-volume NMF code from `gitlab.com/ngillis/nmfbook/`, and used a rank-100 NMF to obtain $U \in \mathbb{R}^{361 \times 100}$, a nonnegative sparse matrix (in fact, 85% of the entries are equal to zero); see Figure 4 (a) for an illustration. Further compressing the NMF factor U using the TSVD does not work as Figure 3 (a) shows, with a relative error larger than 70%; see also Figure 4 (b). This is because NMF tends to generate factors U whose singular values are all large. However, ReLU-NMD does not have this limitation: it can approximate well such sparse full-rank matrices (e.g., the identity matrix [3]). Denoting $\hat{U} = \max(0, \Theta)$ the approximation matrix obtained by a ReLU-NMD algorithm, we first evaluate the relative error as in (10) between U and $\hat{U}$. Results are displayed for increasing values of the rank in Figure 3 (a). We also evaluate the error of the compressed NMF

$$e_{NMF} = \min_{\hat{V} \geq 0} \frac{\|X - \max(0, \hat{U})\hat{V}\|_F}{\|X\|_F}, \qquad (12)$$

see Figure 3 (b). In these experiments, we fix a time limit of 20 seconds. For such data sets, it appears that A-NMD reaches the best solution. For example, with $r = 20$, it is able to reach almost the same accuracy on the NMF problem as the original factor of size 100. Figure 4 shows an example of a rank $r = 20$ reconstruction of the original rank $r = 100$ NMF factor.



(a) Error on U.



(b) Error on X; see (12).

Fig. 3. Compression of a 361-by-100 NMF basis, U, of the CBCL data set. Image (a) shows the error on the NMF basis $U \geq 0$. Image (b) shows the NMF error after U is replaced by its approximation; see (12).

6. CONCLUSION

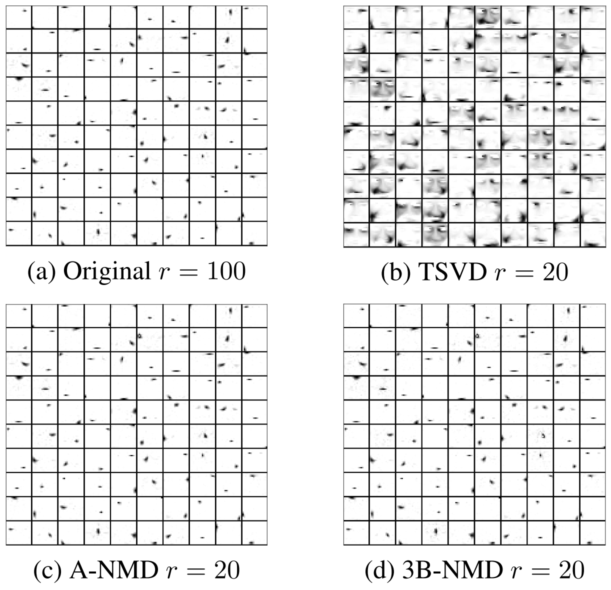
In this paper, we have proposed new algorithms for ReLU-NMD, one accelerating the naive algorithm by Saul (A-NMD), the other reparametrizing the low-rank variable to reduce the computational cost (3B-NMD). We showed their efficiency compared to the state of the art on synthetic and real-world data sets.

7. REFERENCES

[1] C. Eckart and G. Young, “The approximation of one matrix by another of lower rank,” *Psychometrika*, vol. 1, no. 3, pp. 211–218, 1936.

[2] N. Gillis, *Nonnegative Matrix Factorization*. SIAM, Philadelphia, 2020.

[3] L. K. Saul, “A nonlinear matrix decomposition for mining the zeros of sparse data,” *SIAM Journal on Mathematics of Data Science*, vol. 4, no. 2, pp. 431–463, 2022.



(a) Original $r = 100$ (b) TSVD $r = 20$ (c) A-NMD $r = 20$ (d) 3B-NMD $r = 20$

Fig. 4. (a) Original factor U of NMF with 100 columns reshaped as facial features, and its rank-20 approximations by (b) TSVD, (c) A-NMD, and (d) 3B-NMD.

[4] L. K. Saul, “A geometrical connection between sparse and low-rank matrices and its application to manifold learning,” *Trans. Mach. Learn. Res.*, 2023.

[5] B. T. Polyak, “Some methods of speeding up the convergence of iteration methods,” *USSR Comput. Math. Math. Phys.*, vol. 4, no. 5, pp. 1–17, 1964.

[6] A. M. S. Ang and N. Gillis, “Accelerating nonnegative matrix factorization algorithms using extrapolation,” *Neural Computation*, vol. 31, pp. 417–439, 2019.

[7] B. Recht, M. Fazel, and P. A. Parrilo, “Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization,” *SIAM Review*, vol. 52, no. 3, pp. 471–501, 2010.

[8] G. A. Watson, “Characterization of the subdifferential of some matrix norms,” *Linear Algebra and its Applications*, vol. 170, pp. 33–45, 1992.

[9] R. Ge, C. Jin, and Y. Zheng, “No spurious local minima in nonconvex low rank problems: A unified geometric analysis,” in *Int. Conf. on Machine Learning*, 2017.

[10] L. Deng, “The MNIST database of handwritten digit images for machine learning research,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.

[11] D. D. Lee and H. S. Seung, “Learning the parts of objects by non-negative matrix factorization,” *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.