

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/393899748>

# Nonlinear Matrix Decomposition with the Sigmoid Function

Conference Paper · August 2025

CITATIONS

0

READS

88

4 authors:



Harrison Nguyen

University of Mons

2 PUBLICATIONS 6 CITATIONS

SEE PROFILE



Atharva Awari

University of Mons

4 PUBLICATIONS 10 CITATIONS

SEE PROFILE



Arnaud Vandaele

University of Mons

38 PUBLICATIONS 219 CITATIONS

SEE PROFILE



Nicolas Gillis

University of Mons

237 PUBLICATIONS 4,933 CITATIONS

SEE PROFILE

# NONLINEAR MATRIX DECOMPOSITION WITH THE SIGMOID FUNCTION

Harrison Nguyen    Atharva Awari    Arnaud Vandaele    Nicolas Gillis

University of Mons

Rue de Houdain 9, 7000 Mons, Belgium

Harrison.NGUYEN@student.umons.ac.be

{AtharvaAbhijit.AWARI, Arnaud.VANDAELE, Nicolas.GILLIS}@umons.ac.be

## ABSTRACT

The nonlinear matrix decomposition (NMD) of a matrix  $X$  consists in finding factors  $W$  and  $H$  such that  $X \approx f(WH)$  where  $f$  is a nonlinear function applied elementwise. This work focuses on the sigmoid function for the choice of  $f$ , and proposes a block coordinate descent (BCD) method to compute the decomposition. We show the ability of our approach to efficiently compress and reconstruct structured matrices, as exemplified with the identity matrix. Additionally, we compare the performance of our method against the truncated singular value decomposition (TSVD) and the NMD with the ReLU function (ReLU-NMD) in tasks such as data compression and matrix completion on real-world datasets, highlighting its ability in capturing nonlinear structures.

**Index Terms**— Nonlinear Matrix Decomposition (NMD), Sigmoid Function, Block Coordinate Descent (BCD), Matrix Completion.

## 1. INTRODUCTION

Low-rank matrix approximations (LRMAs) are essential techniques in data analysis and machine learning, widely used for tasks such as dimensionality reduction, feature extraction, and data compression. The goal of LRMA is to approximate a matrix  $X \in \mathbb{R}^{m \times n}$  by a low-rank matrix  $\tilde{X} \in \mathbb{R}^{m \times n}$  constructed as the product of two factors,  $W \in \mathbb{R}^{m \times r}$  and  $H \in \mathbb{R}^{r \times n}$  where  $r \ll \min(m, n)$ . This is typically formulated as the following problem:

$$\min_{W, H} \|X - WH\|_F^2, \quad (1)$$

where  $\|\cdot\|_F$  denotes the Frobenius norm. Classical examples include the truncated singular value decomposition (TSVD) [1] and nonnegative matrix factorization (NMF) [2]. These linear methods provide effective tools for analyzing structured data; they approximate the columns of  $X$  with linear combinations of basis vectors  $W$ , and NMF imposes nonnegativity constraints on  $W$  and  $H$  for better interpretability.

However, linear models such as the TSVD and NMF can struggle when data exhibit complex, nonlinear structures. To address this limitation, nonlinear matrix decompositions (NMDs) have recently been introduced by Saul [3]. NMDs extend LRMA by incorporating an elementwise nonlinear function,  $f(\cdot)$ , leading to the approximation:

$$X \approx f(WH), \quad (2)$$

where  $f$  allows the model to capture nonlinear relationships in the data. Saul focused on  $f(\cdot) = \text{ReLU}(\cdot) = \max(0, \cdot)$ . Another example is the component-wise square  $f(\cdot) = (\cdot)^2$  [4, 5]. This paper focuses, for the first time, on NMD where the element-wise nonlinearity is defined by the sigmoid function,

$$f(z) = \sigma(z) = \frac{1}{1 + e^{-z}}, \quad (3)$$

which maps any real number  $z$  to the interval  $[0, 1]$ . This bounded range is a key property that makes the sigmoid function particularly well-suited for approximating matrices  $X$  with entries in this interval; note that the model can be adapted to handle any interval, replacing  $\sigma(z)$  by  $(b - a)\sigma(z) + a$  to handle bounded data in the interval  $[a, b]$ . As one of the most well-known activation functions in neural networks, the sigmoid function is widely recognized for its ability to map inputs to a probabilistic interpretation, which is particularly advantageous in applications where binary or categorical data is prevalent. Such matrices commonly appear in fields like recommendation systems, where entries indicate user preferences, social network analysis, where entries represent connections or interactions, and image processing, where data often encodes presence or absence of features or patterns.

The problem we aim to solve can be expressed as:

$$\min_{W, H} \|X - \sigma(WH)\|_F^2, \quad (4)$$

where  $\sigma(\cdot)$  is applied component-wise on  $WH$ . We refer to this problem (4) as  $\sigma$ -NMD. One property of this model is its ability to compress and reconstruct structured matrices, as opposed to linear models such as the TSVD and NMF. For example, consider  $I_n$ , the identity matrix of dimension  $n$ : for any  $\epsilon > 0$  and for any dimension  $n$ , it is possible to construct  $W(\epsilon) \in \mathbb{R}^{n \times 3}$  and  $H(\epsilon) \in \mathbb{R}^{3 \times n}$  such that  $\|I_n - \sigma(W(\epsilon)H(\epsilon))\|_F \leq \epsilon$ . This follows from a construction in [6] where authors show that there exists a rank-3 matrix,  $A$ , such that  $A(i, i) > 0$  for all  $i$  and  $A(i, j) < 0$  for all  $i \neq j$ ; see also [3]. Hence to obtain an arbitrarily good  $\sigma$ -NMD of the identity, it suffices to multiply this matrix by a sufficiently large scalar.

Beyond its ability to compress structured binary matrices, we will show that  $\sigma$ -NMD is effective in predicting missing entries in incomplete data sets by learning the underlying patterns in the observed data. Let us introduce the  $\sigma$ -NMD with missing data: we introduce a binary mask matrix  $P \in \{0, 1\}^{m \times n}$ , where  $P_{ij} = 0$  if the entry  $X_{ij}$  is missing, and  $P_{ij} = 1$  otherwise, and we consider

$$\min_{W, H} \|X \odot P - \sigma(WH) \odot P\|_F^2, \quad (5)$$

where  $\odot$  denotes the element-wise product. The presence of this mask ensures that the optimization only uses the observed data during training, while predictions of the missing entries are obtained from  $\sigma(WH)$ .

**Contribution and outline of the paper** The paper is organized as follows. Section 2 introduces a block coordinate descent (BCD) algorithm to solve  $\sigma$ -NMD (4). In Section 3, we present experimental results and a detailed comparison of our method with the TSVD and ReLU-NMD on tasks such as data compression and matrix completion on various datasets, highlighting its advantages in capturing nonlinear structures. Section 4 concludes the paper and outlines future research directions.

## 2. PROPOSED ALGORITHM

The  $\sigma$ -NMD, as formulated in (4), is inherently nonconvex. Unlike linear decompositions such as the TSVD or NMF, the nonconvexity persists even if one factor ( $W$  or  $H$ ) is fixed. To address this challenge, we draw inspiration from algorithms widely used for matrix factorizations and adopt an alternating optimization scheme that iteratively updates  $W$  and  $H$ . One of the key properties of the problem is its symmetry: solving for  $H$  given  $W$  is equivalent to solving for  $W^\top$  given  $H^\top$  because the approximation  $X \approx \sigma(WH)$  can be rewritten as  $X^\top \approx \sigma(H^\top W^\top)$ . This symmetry enables the design of algorithms that only need to focus on one factor update at a time. Moreover, the problem is separable across rows of  $W$  and columns of  $H$ . Specifically, the objective function can be expressed as:

$$\|X - \sigma(WH)\|_F^2 = \sum_{j=1}^n \|\sigma(WH(:,j)) - X(:,j)\|_2^2. \quad (6)$$

Consequently, updating  $H$  can be decomposed into a series of independent subproblems, one for each column  $H(:,j)$ , defined as:

$$\min_{H(:,j)} \|\sigma(WH(:,j)) - X(:,j)\|_2^2. \quad (7)$$

These subproblems naturally align with an alternating block coordinate descent scheme, where one block is optimized while the others are kept fixed. To solve (7), we use a least squares approach. The problem can be expressed in general form as:

$$\min_{x \in \mathbb{R}^r} \|\sigma(Ax) - b\|_2^2. \quad (8)$$

We will refer to this minimization problem as a *sigmoid Least Squares* ( $\sigma$ -LS) problem. Unfortunately, the presence of the sigmoid function  $\sigma(\cdot)$  introduces nonlinearity which makes the problem hard to solve; in fact, it is NP-hard in general [7]. We employ an iterative strategy to approximate and optimize  $\sigma$ -LS efficiently.

### 2.1. Gradient descent for each block

In the optimization process for  $\sigma$ -NMD, each block (a column of  $H$  or a row of  $W$ ) is updated iteratively using gradient descent. In our case, the objective function is:  $f(x) = \|\sigma(Ax) - b\|_2^2$ , and the gradient with respect to  $x$  is:

$$\nabla f(x) = 2A^\top \left( (\sigma(Ax) - b) \odot \sigma'(Ax) \right), \quad (9)$$

where  $\sigma'(z) = \sigma(z)(1 - \sigma(z))$  is the derivative of the sigmoid function. To ensure fast progress in each update, a distinct step length  $\alpha$

is used for each block. This allows the optimization process to adapt to the varying scales and sensitivities of different blocks, mitigating issues like overshooting or stagnation that can arise from a uniform step size.

The pseudo-code for the alternating scheme addressing the  $\sigma$ -NMD problem (4) is summarized in Algorithm 1. The algorithm alternates between updating the matrix  $H$  (using the procedure `UpdateH` detailed in Algorithm 2) and updating the matrix  $W$  by transposing the input and output matrices. Additionally, the subproblem  $\sigma$ -LS (8) is solved by Algorithm 3.

---

#### Algorithm 1: Alternating scheme for $\sigma$ -NMD

---

```

1: Input:  $X \in \mathbb{R}^{m \times n}$ ,  $W^{(0)} \in \mathbb{R}^{m \times r}$ ,  $H^{(0)} \in \mathbb{R}^{r \times n}$ , maxiter.
2: Output:  $W \in \mathbb{R}^{m \times r}$  and  $H \in \mathbb{R}^{r \times n}$  s.t.  $X \approx \sigma(WH)$ .
3: for  $k = 1, \dots, \text{maxiter}$  do
4:    $H = \text{UpdateH}(X, W, H)$ 
5:    $W = \text{UpdateH}(X^\top, H^\top, W^\top)^\top$ 
6: end for

```

---



---

#### Algorithm 2: UpdateH - Update Procedure for $H$

---

```

1: Input:  $W^{(k)} \in \mathbb{R}^{m \times r}$ ,  $H^{(k)} \in \mathbb{R}^{r \times n}$ ,  $X \in \mathbb{R}^{m \times n}$ 
2: Output:  $H^{(k+1)}$ 
3: for  $i = 1, \dots, r$  do
4:    $sc(i) = \|W(:,i)\|_2$ 
5:    $W_n(:,i) = W(:,i)/sc(i)$ 
6:   for  $j = 1$  to  $n$  do
7:      $H(i,j) = H(i,j) \times sc(i)$ 
8:   end for
9: end for
10: for  $j = 1, \dots, n$  do
11:    $H(:,j) = \sigma\text{-LS}(W_n, X(:,j), H(:,j))$ 
12:   for  $i = 1, \dots, r$  do
13:      $H(i,j) = H(i,j)/sc(i)$ 
14:   end for
15: end for

```

---



---

#### Algorithm 3: $\sigma$ -LS - Gradient Descent

---

```

1: Input:  $A \in \mathbb{R}^{m \times r}$ ,  $b \in \mathbb{R}^m$ ,  $x \in \mathbb{R}^r$ ,  $\alpha \in \mathbb{R}$ .
2: Output:  $x \in \mathbb{R}^r$  solving (8)
3:  $d = -2A^\top \left( (\sigma(Ax) - b) \odot \sigma'(Ax) \right)$ 
4: if  $\sum_i |d_i| \geq 10^{-6}$  then
5:    $\alpha = \text{ComputeStepLength}(A, b, x, d, \alpha)$ 
6:    $x = x + \alpha d$ 
7: end if

```

---

**Step length selection** To ensure efficient convergence and avoid overshooting or undershooting during the update, the function `ComputeStepLength` described in Algorithm 4 is used to determine the step length  $\alpha$ : it is a backtracking line search to satisfy the Armijo-Wolfe conditions [8]. These conditions ensure a balance between sufficient decrease in the objective function and adequate progress along the search direction. The step length  $\alpha$  is computed using a bisection algorithm, which iteratively reduces the interval  $[\alpha_{min}, \alpha_{max}]$  until the Armijo-Wolfe conditions are satisfied.

**Computational cost** Each iteration of Algorithm 1 involves two calls to `UpdateH` (once for  $H$  and once for  $W$ ), whose computational complexity is dominated by the updates of the  $n$  columns of  $H$  via the  $\sigma$ -LS subroutine. This subroutine performs a single gradient

**Algorithm 4:** Compute Step Length in  $\sigma$ -LS

---

```

1: Input:  $A \in \mathbb{R}^{m \times r}$ ,  $b \in \mathbb{R}^m$ ,  $x \in \mathbb{R}^r$ ,  $\alpha_0 \in \mathbb{R}$ ,  $(\beta_1, \beta_2)$ 
   [default =  $(\beta_1, \beta_2) = (10^{-4}, 0.9)$ ].
2: Output:  $\alpha \in \mathbb{R}$ 
3:  $\alpha = \alpha_0$ 
4:  $\alpha_{min} = 0$ 
5:  $\alpha_{max} = \infty$ 
6: repeat
7:    $f_0 = \|\sigma(Ax) - b\|_2$ 
8:    $f_1 = \|\sigma(A(x + \alpha d)) - b\|_2$ 
9:    $g_0 = 2A^\top ((\sigma(Ax) - b) \odot \sigma'(Ax))$ 
10:   $g_1 = 2A^\top ((\sigma(A(x + \alpha d)) - b) \odot \sigma'(A(x + \alpha d)))$ 
11:  if  $f_1 > f_0 + \alpha\beta_1 d^\top g_0$  then
12:     $\alpha_{max} = \alpha$ 
13:     $\alpha = \frac{\alpha_{min} + \alpha_{max}}{2}$ 
14:  else if  $d^\top g_1 < \beta_2 d^\top g_0$  then
15:     $\alpha_{min} = \alpha$ 
16:    if  $\alpha_{max} < \infty$  then
17:       $\alpha = \frac{\alpha_{min} + \alpha_{max}}{2}$ 
18:    else
19:       $\alpha = 2\alpha$ 
20:    end if
21:  end if
22: until  $f_1 \leq f_0 + \alpha\beta_1 d^\top g_0$  and  $d^\top g_1 \geq \beta_2 d^\top g_0$ 

```

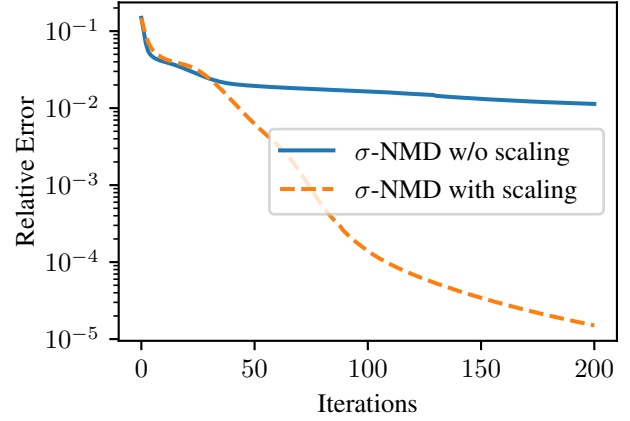
---

descent step with a backtracking line search, where the number of steps is typically small in practice ( $\approx 1.23$  on average). For the update of the columns of  $H$ , each gradient step requires matrix-vector products with a computational cost of  $\mathcal{O}(mr)$  operations. The same analysis holds for  $W$  where the roles of  $m$  and  $n$  are exchanged. As a result, the overall computational complexity of our proposed algorithm is  $\mathcal{O}(mnr)$  operations per iteration. This is the same cost per iteration as first-order methods for standard low-rank matrix approximations such as NMF.

## 2.2. Scaling for faster convergence

To address potential inconsistencies arising from significant differences in the column norms of  $W$  when updating  $H$ , we incorporate a scaling procedure into the optimization process. This approach not only reduces imbalances that could reduce the effectiveness of the algorithm and hinder convergence, but also improves the conditioning of the problem. Specifically, the Hessian of the optimization problem is related to  $W^\top W$  (this is the Hessian in the linear case), which tends to be better conditioned when the columns of  $W$  are normalized. By normalizing  $W$ , the scaling procedure ensures a more stable and efficient optimization process. The proposed scaling procedure consists of two key steps, both of which are explicitly integrated into the UpdateH procedure (see Algorithm 2):

1. *Pre-scaling:* Before updating  $H$ , each column of  $W$  is normalized by its  $\ell_2$ -norm. To preserve the overall decomposition, the corresponding rows of  $H$  are scaled proportionally by the same factors. This ensures that the product  $WH$  remains invariant under the scaling transformation, maintaining the integrity of the factorization.
2. *Post-scaling:* After the optimization step, where updates to  $H$  are computed, the earlier scaling is reversed. This involves dividing the rows of  $H$  by their respective scaling factors.



**Fig. 1.** Average relative error  $\frac{\|X - \sigma(WH)\|_F}{\|X\|_F}$  of  $\sigma$ -NMD with scaling and without scaling for 10 randomly generated matrices  $X \in \mathbb{R}^{200 \times 200}$  with  $r = 5$  and  $maxiter = 200$ .

This operation restores  $H$  to its proper scale, ensuring that the updated matrices are correctly normalized and that the final decomposition accurately represents the input data.

Figure 1 highlights the impact of incorporating scaling into the  $\sigma$ -NMD algorithm evaluated on synthetic data  $X = \sigma(W_t H_t) \in \mathbb{R}^{200 \times 200}$ , where  $W_t \in \mathbb{R}^{200 \times 5}$  and  $H_t \in \mathbb{R}^{5 \times 200}$  are generated with uniformly random entries using the Python function `numpy.random.rand`. We observe that the addition of scaling contributes to faster convergence by addressing large variations in the norms of the columns of  $W$  and rows of  $H$ .

## 2.3. Initializations

We adopt two initialization strategies:

1. *Random initialization:* The entries of  $W$  and  $H$  are independently sampled from a standard Gaussian distribution  $\mathcal{N}(0, 1)$ .
2. *TSVD-based initialization:* In this approach, we aim to solve the following problem using the TSVD:

$$\|f^{-1}(X) - WH\|_F^2,$$

where  $f^{-1}$  is the inverse of the sigmoid function. Since the sigmoid function is bijective from  $(-\infty, +\infty)$  to  $(0, 1)$ , its inverse is defined for all  $y \in (0, 1)$  as :

$$f^{-1}(y) = -\log\left(\frac{1}{y} - 1\right).$$

However, this inverse is not defined at the endpoints  $y \in \{0, 1\}$ , which poses a problem when  $X$  is a binary matrix. To address this issue, we approximate  $f^{-1}(X)$  by a linear transformation:

$$f^{-1}(X) \approx 2X - 1,$$

which maps 0 to  $-1$  and 1 to  $+1$ , preserving the sign structure of the inverse sigmoid. We then compute the TSVD of the transformed matrix:

$$2X - 1 \approx U\Sigma V^\top,$$

where  $U \in \mathbb{R}^{m \times r}$ ,  $\Sigma \in \mathbb{R}^{r \times r}$ , and  $V \in \mathbb{R}^{n \times r}$ . The initialization of the factors  $W$  and  $H$  is then performed using:

$$W = U\sqrt{\Sigma}, \quad H = \sqrt{\Sigma}V^\top.$$

### 3. NUMERICAL EXPERIMENTS

The goal of these experiments is to compare the performance of our method against the performance of the TSVD, and NMD with the ReLU function (ReLU-NMD). For the TSVD, we use the `numpy.linalg.svd` function for complete matrices, and for matrices with missing values, we use the weighted low-rank approximation (WLRA) algorithm described in [9] and available from <https://gitlab.com/ngillis/nmfbook.git>. For ReLU-NMD, we use the recent extrapolated block coordinate descent algorithm (eBCD-NMD) described in [10] and available from <https://github.com/giovanniseraghiti/ReLU-NMD>. The codes are available from <https://github.com/Harri-N/sigma-NMD> and implemented in Python.

#### 3.1. Approximation of binary matrices

We evaluate our algorithm on four real datasets used in [11] and summarized in Table 1.

Dataset	Size	# observed	%1s
zoo	101 × 17	1717	44.3
heart	242 × 22	5324	34.4
lymp	148 × 44	6512	29.0
apb	105 × 105	11 025	8.0

**Table 1.** Summary of binary real world datasets

To assess the quality of the approximation, we use two main metrics: the relative error and the error rate. Let  $\tilde{X}$  denote the approximation of the matrix  $X$ . For TSVD,  $\tilde{X} = WH$ , for eBCD-NMD,  $\tilde{X} = \max(0, WH)$ , and for  $\sigma$ -NMD,  $\tilde{X} = \sigma(WH)$ .

**Relative Error** The relative error quantifies the overall deviation of the approximation  $\tilde{X}$  from the original matrix  $X$ , normalized by the Frobenius norm of  $X$ . It is defined as:

$$\text{Relative Error} = \frac{\|X - \tilde{X}\|_F}{\|X\|_F}.$$

**Error Rate** The error rate evaluates the algorithm’s ability to reconstruct binary matrices accurately. It is defined as the percentage of misclassifications after rounding the values of  $\tilde{X}$  to either 0 or 1. Specifically, each element of  $\tilde{X}$  is rounded as follows:

$$\tilde{X}_{ij} = \begin{cases} 1, & \text{if } \tilde{X}_{ij} \geq 0.5, \\ 0, & \text{otherwise.} \end{cases}$$

The error rate is then computed as:

$$\text{Error Rate (\%)} = \frac{\text{Number of misclassified elements}}{\text{Total number of elements}} \times 100.$$

In our experiments, we established stopping criteria to ensure convergence. Specifically, we stop the algorithm at iteration  $i + 10$  if the following condition is met:

$$\text{relerr}(i + 10) \geq \alpha \text{relerr}(i) \text{ or } \text{relerr}(i + 10) \leq \epsilon,$$

where  $\text{relerr}(k)$  denotes the relative error at iteration  $k$ . For all our experiments, we set  $\alpha = 0.9999$  and  $\epsilon = 10^{-4}$ . This ensures that the algorithm halts either when the relative improvement becomes negligible over ten iterations or when the error is sufficiently small.

Table 2 presents the relative error and error rate, respectively, for the  $\sigma$ -NMD, eBCD-NMD and TSVD methods applied to the datasets summarized in Table 1. For  $\sigma$ -NMD and eBCD-NMD, we include results for both random and TSVD-based initializations. For the random case, we perform 10 different runs with different random initializations and report the average over these runs.

These results show that, in most cases,  $\sigma$ -NMD outperforms eBCD-NMD and TSVD in terms of both relative error and error rate on compressing binary matrices. Additionally, initializing  $\sigma$ -NMD with SVD improves the performance, on average, but the difference is not significant.

#### 3.2. Matrix completion

We evaluate the ability of  $\sigma$ -NMD to predict missing entries in incomplete datasets. We introduce a new metric to assess the prediction of missing values defined as the root mean squared error (RMSE) over the missing entries, given by:

$$\text{RMSE}_{\bar{\Omega}} = \sqrt{\frac{\sum_{i,j} (1 - P_{ij}) (X_{ij} - \tilde{X}_{ij})^2}{\sum_{i,j} 1 - P_{ij}}},$$

where  $P_{ij} = 0$  if the entry is missing, and  $P_{ij} = 1$  otherwise. This metric ensures that the evaluation reflects the algorithm’s ability to predict the missing data, rather than its performance on the observed data.

In this case, the algorithms are tested on their ability to predict these missing values, and both the RMSE and the error rate are computed exclusively on the missing entries, rather than across all input data.

For our experiments, we use a subset of 243 images from the CBCL dataset (to keep the time for the numerical experiments reasonable, since we will average the results over 10 runs, using 3 ranks and 5 levels of missing data for each data set), a collection of 2429 grayscale facial images with  $19 \times 19$  pixels each. To simulate incomplete binary data, we first normalize the input matrix by dividing all entries by the largest value, then apply rounding to obtain binary values. A subset of the entries is then randomly masked to simulate missing data.

In the case of matrix completion, we use the WLRA algorithm to perform the TSVD. This algorithm is initialized randomly, making it nondeterministic. Therefore, all reported results are averaged over 10 runs. Table 3 reports the relative error on the observed entries in addition to the RMSE and error rate on the missing ones.

Across all configurations, our proposed method consistently achieves the lowest reconstruction error on the observed entries, particularly when initialized with TSVD. Regarding the performance on missing entries,  $\sigma$ -NMD also shows competitive performance, especially in low-rank settings. At rank  $r = 2$ , it outperforms all baselines in both RMSE and error rate, regardless of the percentage of missing data. At rank  $r = 5$ , it remains competitive and often achieves the best performance, especially when the proportion of missing values is moderate (10–30%). At rank  $r = 10$ , TSVD and eBCD-NMD catch up in terms of RMSE or error rate.

Data	rank	$\sigma$ -NMD (TSVD init.)	$\sigma$ -NMD (random init.)	eBCD-NMD (TSVD init.)	eBCD-NMD (random init.)	TSVD
1	2	<b>38.1</b> - 11.9	38.2 - 12.6	45.9 - 11.4	45.9 - <b>11.3</b>	48.5 - 13.7
	5	<b>5.13</b> - <b>0.12</b>	9.52 - 0.44	16.5 - 0.93	16.9 - 0.97	30.0 - 3.96
	10	<b>0.02</b> - <b>0.0</b>	2.24 - 0.05	0.87 - <b>0.0</b>	0.74 - <b>0.0</b>	15.2 - 0.35
2	2	<b>59.9</b> - 20.0	60.6 - 21.2	62.6 - <b>19.0</b>	62.8 - 19.3	63.7 - 19.3
	5	<b>30.4</b> - 3.91	30.6 - <b>3.81</b>	42.5 - 8.41	42.4 - 8.44	49.6 - 10.7
	10	<b>7.75</b> - <b>0.21</b>	12.2 - 0.52	18.8 - 1.05	18.7 - 1.07	36.1 - 4.04
3	2	61.5 - <b>16.7</b>	<b>61.3</b> - 16.8	64.2 - 16.9	64.2 - 16.9	64.9 - 17.1
	5	<b>35.7</b> - <b>4.64</b>	37.2 - 5.1	48.8 - 8.66	48.8 - 8.63	54.0 - 10.7
	10	<b>10.0</b> - <b>0.29</b>	14.1 - 0.58	22.2 - 1.0	22.9 - 1.12	41.0 - 4.04
4	2	77.0 - 6.41	77.1 - 6.31	<b>76.1</b> - <b>6.15</b>	76.8 - 6.26	82.8 - 6.89
	5	<b>38.1</b> - <b>1.24</b>	39.6 - 1.34	52.2 - 2.66	51.9 - 2.65	75.8 - 6.13
	10	17.2 - 0.24	23.3 - 0.44	8.13 - 0.03	<b>6.48</b> - <b>0.02</b>	66.2 - 4.23

**Table 2.** Relative Error (%) and Error Rate (%) of  $\sigma$ -NMD, eBCD-NMD, TSVD on zoo (1), heart (2), lymf (3) and apb (4) datasets.

missing (%)	1's (%)	rank	$\sigma$ -NMD (TSVD init.)	$\sigma$ -NMD (random init.)	eBCD-NMD (TSVD init.)	eBCD-NMD (random init.)	TSVD
10	54	2	<b>44.5</b> - <b>0.34</b> - <b>16.3</b>	<b>44.5</b> - <b>0.34</b> - <b>16.3</b>	46.0 - 0.35 - 17.4	47.0 - 0.35 - 17.8	46.1 - <b>0.34</b> - 17.0
		5	<b>35.0</b> - <b>0.31</b> - <b>12.0</b>	35.2 - <b>0.31</b> - 12.5	40.3 - <b>0.31</b> - 13.2	41.6 - 0.32 - 13.5	40.3 - <b>0.31</b> - 13.1
		10	<b>21.7</b> - 0.33 - 11.5	22.1 - 0.33 - 11.6	36.0 - <b>0.29</b> - <b>10.8</b>	37.7 - 0.31 - 11.9	36.0 - <b>0.29</b> - 10.9
20	54	2	<b>44.5</b> - <b>0.34</b> - <b>16.0</b>	<b>44.5</b> - <b>0.34</b> - <b>16.0</b>	46.0 - <b>0.34</b> - 17.1	49.4 - 0.37 - 18.9	46.1 - <b>0.34</b> - 16.5
		5	<b>34.4</b> - 0.32 - <b>12.6</b>	34.5 - 0.32 - <b>12.6</b>	40.2 - <b>0.31</b> - 13.2	44.4 - 0.35 - 15.7	40.2 - <b>0.31</b> - 13.0
		10	<b>20.8</b> - 0.33 - 11.6	21.2 - 0.34 - 11.7	35.7 - <b>0.29</b> - 11.1	41.0 - 0.35 - 16.0	35.9 - <b>0.29</b> - <b>10.7</b>
30	54	2	<b>44.4</b> - <b>0.34</b> - <b>16.1</b>	<b>44.4</b> - <b>0.34</b> - 16.2	46.1 - <b>0.34</b> - 16.7	53.2 - 0.4 - 22.2	46.1 - <b>0.34</b> - 16.6
		5	<b>33.9</b> - 0.32 - <b>12.5</b>	34.0 - 0.33 - 12.8	40.1 - <b>0.31</b> - 12.9	48.7 - 0.38 - 19.7	40.1 - <b>0.31</b> - 12.9
		10	<b>19.4</b> - 0.35 - 12.3	20.1 - 0.35 - 12.6	35.5 - <b>0.29</b> - <b>11.0</b>	45.8 - 0.39 - 22.0	35.5 - <b>0.29</b> - <b>11.0</b>
40	54	2	<b>44.2</b> - <b>0.34</b> - 16.4	<b>44.2</b> - <b>0.34</b> - <b>16.3</b>	46.0 - <b>0.34</b> - 16.7	57.9 - 0.43 - 29.1	46.1 - <b>0.34</b> - 16.8
		5	<b>32.6</b> - 0.34 - 13.5	33.0 - 0.34 - 13.7	39.8 - <b>0.31</b> - <b>13.0</b>	53.9 - 0.42 - 27.9	39.9 - <b>0.31</b> - 13.1
		10	<b>18.0</b> - 0.36 - 13.2	18.9 - 0.36 - 13.6	35.0 - <b>0.3</b> - 11.5	51.1 - 0.44 - 32.1	35.1 - <b>0.3</b> - <b>11.4</b>
50	54	2	<b>43.9</b> - <b>0.34</b> - <b>16.5</b>	<b>43.9</b> - <b>0.34</b> - <b>16.5</b>	46.2 - <b>0.34</b> - 16.6	63.9 - 0.48 - 43.5	46.0 - <b>0.34</b> - 16.7
		5	<b>31.7</b> - 0.34 - 13.7	32.1 - 0.35 - 14.1	39.9 - <b>0.31</b> - 13.2	60.5 - 0.47 - 41.5	39.8 - <b>0.31</b> - <b>13.1</b>
		10	<b>16.1</b> - 0.37 - 14.1	17.6 - 0.37 - 14.6	34.7 - <b>0.3</b> - <b>11.7</b>	57.5 - 0.5 - 42.2	34.6 - 0.31 - 11.8

**Table 3.** Relative Error (%) on observed values, RMSE and Error Rate (%) on missing entries of  $\sigma$ -NMD, eBCD-NMD, TSVD on the CBCL dataset.

missing (%)	1's (%)	rank	$\sigma$ -NMD (TSVD init.)	$\sigma$ -NMD (random init.)	eBCD-NMD (TSVD init.)	eBCD-NMD (random init.)	TSVD
10	14	2	<b>77.5</b> - <b>0.31</b> - 12.4	78.1 - <b>0.31</b> - 12.6	79.5 - <b>0.31</b> - <b>12.2</b>	79.8 - <b>0.31</b> - 12.3	79.5 - <b>0.31</b> - 12.3
		5	61.5 - <b>0.28</b> - <b>9.74</b>	<b>61.4</b> - <b>0.28</b> - 9.78	72.6 - 0.29 - 11.2	72.8 - 0.29 - 11.3	72.7 - 0.29 - 11.1
		10	<b>45.1</b> - 0.32 - 11.0	45.8 - 0.32 - 10.9	66.2 - <b>0.27</b> - 10.2	66.2 - <b>0.27</b> - 10.2	66.6 - 0.28 - <b>10.1</b>
20	14	2	<b>77.2</b> - <b>0.31</b> - <b>12.5</b>	<b>77.7</b> - <b>0.31</b> - 12.6	79.4 - <b>0.31</b> - <b>12.5</b>	80.4 - <b>0.31</b> - 12.8	79.3 - <b>0.31</b> - 12.6
		5	60.7 - <b>0.29</b> - <b>10.2</b>	<b>60.4</b> - <b>0.29</b> - <b>10.2</b>	72.4 - <b>0.29</b> - 11.2	73.5 - <b>0.29</b> - 11.8	72.1 - <b>0.29</b> - 11.3
		10	<b>43.4</b> - 0.33 - 11.1	44.2 - 0.33 - 11.3	66.2 - <b>0.28</b> - <b>10.1</b>	67.2 - <b>0.28</b> - 10.8	66.1 - <b>0.28</b> - <b>10.1</b>
30	15	2	<b>77.3</b> - <b>0.31</b> - <b>12.5</b>	<b>77.6</b> - <b>0.31</b> - <b>12.5</b>	79.4 - <b>0.31</b> - 12.9	81.4 - 0.32 - 13.6	79.4 - <b>0.31</b> - <b>12.5</b>
		5	59.6 - 0.3 - <b>10.6</b>	<b>59.5</b> - 0.3 - <b>10.6</b>	72.2 - <b>0.29</b> - 11.6	75.0 - 0.3 - 12.9	72.1 - <b>0.29</b> - 11.3
		10	<b>40.7</b> - 0.34 - 11.8	41.7 - 0.34 - 12.0	65.9 - <b>0.28</b> - 10.4	69.0 - 0.29 - 12.3	65.7 - <b>0.28</b> - <b>10.2</b>
40	15	2	<b>77.5</b> - <b>0.31</b> - <b>12.3</b>	<b>77.8</b> - <b>0.31</b> - 12.5	79.3 - <b>0.31</b> - 12.6	82.8 - 0.32 - 13.9	79.4 - <b>0.31</b> - 12.5
		5	59.3 - 0.31 - 11.1	<b>58.7</b> - 0.3 - <b>10.9</b>	72.0 - <b>0.29</b> - 11.5	76.8 - 0.31 - 13.5	71.9 - <b>0.29</b> - 11.4
		10	<b>38.1</b> - 0.35 - 12.4	39.4 - 0.35 - 12.8	65.3 - <b>0.28</b> - <b>10.3</b>	71.1 - 0.3 - 13.3	65.3 - <b>0.28</b> - 10.4
50	15	2	<b>77.1</b> - <b>0.31</b> - <b>12.6</b>	<b>77.2</b> - <b>0.31</b> - 12.7	79.5 - <b>0.31</b> - 12.9	84.9 - 0.33 - 14.5	79.5 - <b>0.31</b> - 12.7
		5	58.2 - 0.32 - 11.9	<b>57.4</b> - 0.32 - 11.8	71.7 - <b>0.29</b> - 11.7	79.3 - 0.32 - 14.3	71.9 - 0.3 - <b>11.5</b>
		10	<b>35.4</b> - 0.36 - 13.2	37.9 - 0.36 - 13.8	64.7 - <b>0.29</b> - 10.9	74.0 - 0.32 - 14.3	64.8 - <b>0.29</b> - <b>10.7</b>

**Table 4.** Relative Error (%) on observed values, RMSE and Error Rate (%) on missing entries of  $\sigma$ -NMD, eBCD-NMD, TSVD on the TDT2 dataset.

In addition to the CBCL dataset, we conducted experiments on a subset of the TDT2 dataset [12] consisting of 382 documents represented by 299 words. As shown in Table 4, the results on TDT2 follow the same trends observed with the CBCL dataset.  $\sigma$ -NMD generally outperforms other methods, achieving the lowest RMSE and error rate on missing entries at lower ranks, and the best reconstruction error on observed data across most settings, regardless of the proportion of missing values.

### 3.3. Reconstruction of the identity matrix

Table 5 evaluates the ability of  $\sigma$ -NMD to reconstruct the identity matrix of dimension  $n$ ,  $X = I_n$ , for various matrix sizes  $n$  and ranks  $r$ . The table reports the percentage of successful reconstructions, defined as instances where the error rate is exactly zero, out of 100 trials.

The results show that  $\sigma$ -NMD achieves near-perfect reconstruction of the identity matrix for ranks  $r = 4$  and  $r = 5$ , with a success rate above 85% for all matrix sizes. For  $r = 5$ , the performance remains above 95% even for larger matrices ( $n = 20$  and  $n = 25$ ). In contrast,  $r = 3$  yields more variable results, with success rates dropping to 18% for  $n = 25$ . This highlights the complexity of the problem, and the fact that  $\sigma$ -NMD can be stuck in spurious local minima – Recall that the identity of any dimension can be approximated up to any precision by  $\sigma$ -NMD of rank 3; see Section 1.

Compared to eBCD-NMD and the TSVD,  $\sigma$ -NMD demonstrates superior robustness and accuracy. As expected, the TSVD cannot reconstruct the identity matrix except for the trivial case  $r = n = 5$ . eBCD-NMD performs well for small  $n$  and higher ranks, but its performance is less consistent than that of  $\sigma$ -NMD, particularly in settings with limited rank or larger matrix sizes.

Overall, these results confirm that  $\sigma$ -NMD is highly effective at capturing structured patterns.

$n$	$r$	$\sigma$ -NMD (TSVD)	$\sigma$ -NMD (rand)	eBCD-NMD (TSVD)	eBCD-NMD (rand)	TSVD
5	3	100	94	0	100	0
	4	100	100	0	100	0
	5	100	100	100	100	100
10	3	0	62	0	2	0
	4	100	99	0	99	0
	5	100	100	0	100	0
15	3	100	82	0	0	0
	4	100	93	0	21	0
	5	100	99	0	99	0
20	3	100	69	0	0	0
	4	100	89	0	1	0
	5	100	95	0	58	0
25	3	0	18	0	0	0
	4	100	85	0	0	0
	5	100	95	0	38	0

**Table 5.** Number of times, out of 100 trials, when the error rate of  $\sigma$ -NMD, eBCD-NMD and TSVD is equal to zero for  $X = I_n$ . Note that algorithms which are initialized using the TSVD, and the TSVD itself, have value 0 or 100 since they are deterministic.

## 4. CONCLUSION

In this paper, we proposed  $\sigma$ -NMD (4), a new nonlinear matrix decomposition model using the sigmoid function.  $\sigma$ -NMD can approx-

imate matrices whose entries are in a given interval (we focused on  $[0, 1]$ , but the proposed algorithm can be easily adapted) and is able to approximate well-structured matrices (such as the identity) as well as identify nonlinear structures. We then designed an algorithm for  $\sigma$ -NMD using an alternating block coordinate descent scheme with gradient-based updates. We also proposed a scaling strategy to accelerate convergence, as well as a clever initialization scheme based on the TSVD. Experimental results show that  $\sigma$ -NMD outperforms TSVD and ReLU-NMD in compressing and completing binary matrices, achieving lower errors. It also has significantly better performance on the identity matrix. Future work will focus on accelerating the algorithm and extending it to other nonlinear decomposition tasks.

## 5. 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] 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.
- [3] Lawrence K Saul, “A nonlinear matrix decomposition for minimizing the zeros of sparse data,” *SIAM J. Math. Data Sci.*, vol. 4, no. 2, pp. 431–463, 2022.
- [4] Lorenzo Loconte, Nicola Di Mauro, Robert Peharz, and Antonio Vergari, “How to turn your knowledge graph embeddings into generative models,” *Advances in Neural Information Processing Systems*, vol. 36, pp. 77713–77744, 2023.
- [5] Joakim Lefebvre, Arnaud Vandaele, and Nicolas Gillis, “Component-wise squared factorization,” in *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2024, pp. 1–6.
- [6] Philippe Delsarte and Yves Kamp, “Low rank matrices with a given sign pattern,” *SIAM Journal on Discrete Mathematics*, vol. 2, no. 1, pp. 51–63, 1989.
- [7] Jiří Šíma, “Minimizing the quadratic training error of a sigmoid neuron is hard,” in *Algorithmic Learning Theory: 12th International Conference, ALT 2001 Washington, DC, USA, November 25–28, 2001 Proceedings 12*. Springer, 2001, pp. 92–105.
- [8] Jorge Nocedal and Stephen J Wright, *Numerical optimization*, Springer, New York, 1999.
- [9] Nicolas Gillis, *Nonnegative Matrix Factorization*, SIAM, Philadelphia, 2020.
- [10] Nicolas Gillis, Margherita Porcelli, and Giovanni Seraghi, “An extrapolated and provably convergent algorithm for nonlinear matrix decomposition with the ReLU function,” 2025.
- [11] Oktay Günlük, Raphael Andreas Hauser, and Réka Ágnes Kovács, “Binary matrix factorization and completion via integer programming,” *Mathematics of Operations Research*, vol. 49, no. 2, pp. 1278–1302, 2024.
- [12] Christopher Cieri, David Graff, Mark Liberman, Nii Martey, and Stephanie Strassel, “The TDT-2 text and speech corpus,” *Proceedings of DARPA Broadcast News Workshop*, 08 2000.