Pattern Recognition Letters 104 (2018) 1–7

Contents lists available at ScienceDirect

Pattern Recognition Letters

journal homepage: www.elsevier.com/locate/patrec

Randomized nonnegative matrix factorization

N. Benjamin Erichson [a,*], Ariana Mendible [a], Sophie Wihlborn [b], J. Nathan Kutz [a]

[a] Department of Applied Mathematics, University of Washington, Seattle, USA
[b] Fidelity International, London, UK

ARTICLE INFO

Article history:
Received 6 July 2017
Available online 17 January 2018

Keywords:
NMF
Randomized algorithm
Dimension reduction

ABSTRACT

Nonnegative matrix factorization (NMF) is a powerful tool for data mining. However, the emergence of 'big data' has severely challenged our ability to compute this fundamental decomposition using deterministic algorithms. This paper presents a randomized hierarchical alternating least squares (HALS) algorithm to compute the NMF. By deriving a smaller matrix from the nonnegative input data, a more efficient nonnegative decomposition can be computed. Our algorithm scales to big data applications while attaining a near-optimal factorization, i.e., the algorithm scales with the target rank of the data rather than the ambient dimension of measurement space. The proposed algorithm is evaluated using synthetic and real world data and shows substantial speedups compared to deterministic HALS.

1. Introduction

Techniques for dimensionality reduction, such as principal component analysis (PCA), are essential to the analysis of high-dimensional data. These methods take advantage of redundancies in the data in order to find a low-rank and parsimonious model describing the underlying structure of the input data. Indeed, at the core of machine learning is the assumption that low-rank structures are embedded in high-dimensional data [34]. Dimension reduction techniques find basis vectors which represent the data as a linear combination in lower-dimensional space. This enables the identification of key features and efficient analysis of high-dimensional data.

A significant drawback of PCA and other commonly-used dimensionality reduction techniques is that they permit both positive and negative terms in their components. In many data analysis applications, negative terms fail to hold physically meaningful interpretation. For example, images are represented as a grid of nonnegative pixel intensity values. In this context, the negative terms in principal components have no interpretation.

To address this problem, researchers have proposed restricting the set of basis vectors to nonnegative terms [23,28]. The paradigm is called nonnegative matrix factorization (NMF) and it has emerged as a powerful dimension reduction technique. This versatile tool allows computation of sparse (parts-based) and physically meaningful factors that describe coherent structures within the data. Prominent applications of NMF are in the areas of image processing, information retrieval and gene expression analysis, see for instance the surveys by Berry et al. [2] and Gillis [13]. However, NMF is computationally intensive and becomes infeasible for massive data. Hence, innovations that reduce computational demands are increasingly important in the era of 'big data'.

Randomized methods for linear algebra have been recently introduced to ease the computational demands posed by classical matrix factorizations [8,25]. Inspired by these ideas, Tepper and Sapiro [32] proposed compressed accelerated NMF algorithms based on the idea of bilateral random projections [40]. While these compressed algorithms reduce the computational load considerably, they often fail to converge in many experiments.

We follow the probabilistic approach for matrix approximations formulated by Halko et al. [17]. Specifically, we propose a randomized hierarchical alternating least squares (HALS) algorithm to compute the NMF. We demonstrate that the randomized algorithm eases the computational challenges posed by massive data, assuming that the input data feature low-rank structure. Experiments show that our algorithm is reliable and attains a near-optimal factorization. Further, this manuscript is accompanied by the open-software package *ristretto*, written in *Python*, which allows the reproduction of all results (GIT repository: https://github.com/erichson/ristretto).

The manuscript is organized as follows: First, Section 2 briefly reviews the NMF as well as the basic concept of randomized matrix algorithms. Then, Section 3 describes a randomized variant of the HALS algorithm. This is followed by an empirical evaluation in Section 4, where synthetic and real world data are used to demonstrate the performance of the algorithm. Finally, Section 5 concludes the manuscript.

* Corresponding author.
E-mail addresses: erichson@uw.edu, nbe@st-andrews.ac.uk (N.B. Erichson).

https://doi.org/10.1016/j.patrec.2018.01.007
0167-8655/

2. Background

2.1. Low-rank matrix factorization

Low-rank approximations are fundamental and widely used tools for data analysis, dimensionality reduction, and data compression. The goal of these methods is to find two matrices of much lower rank that approximate a high-dimensional matrix $\mathbf{X}$:

$$\underset{m\times n}{\mathbf{X}} \approx \underset{m\times k}{\mathbf{W}} \underset{k\times n}{\mathbf{H}}. \tag{1}$$

The target rank of the approximation is denoted by k, an integer between 1 and $\min\{m, n\}$. A ubiquitous example of these tools, the singular value decomposition (SVD), finds the exact solution to this problem in a least-square sense [9]. While the optimality property of the SVD and similar methods is desirable in many scientific applications, the resulting factors are not guaranteed to be physically meaningful in many others. This is because the SVD imposes orthogonality constraints on the factors, leading to a holistic, rather than parts-based, representation of the input data. Further, the basis vectors in the SVD and other popular decompositions are mixed in sign.

Thus, it is natural to formulate alternative factorizations which may not be optimal in a least-square sense, but which may preserve useful properties such as sparsity and nonnegativity. Such properties are found in the NMF.

2.2. Nonnegative matrix factorization

The roots of NMF can be traced back to the work by Paatero and Tapper [28]. Lee and Seung [23] independently introduced and popularized the concept of NMF in the context of psychology several years later.

Formally, the NMF attempts to solve Eq. (1) with the additional nonnegativity constraints: $\mathbf{W} \geq 0$ and $\mathbf{H} \geq 0$. These constraints enforce that the input data are expressed as an additive linear combination. This leads to sparse, parts-based features appearing in the decomposition, which have an intuitive interpretation. For example, NMF components of a face image dataset reveal individual nose and mouth features, whereas PCA components yield holistic features, known as 'eigenfaces'.

Though NMF bears the desirable property of interpretability, the optimization problem is inherently nonconvex and ill-posed. In general, no convexification exists to simplify the optimization, meaning that no exact or unique solution is guaranteed [14]. Different NMF algorithms, therefore, can produce distinct decompositions that minimize the objective function.

We refer the reader to [2,13,24] for a comprehensive discussion of the NMF and its applications. There are two main classes of NMF algorithms [14], discussed in the following.

2.2.1. Standard nonlinear optimization schemes

Traditionally, the challenge of finding the nonnegative factors is formulated as the following optimization problem:

$$\begin{aligned} &\text{minimize} \quad f(\mathbf{W},\mathbf{H}) = \|\mathbf{X}-\mathbf{W}\mathbf{H}\|_F^2 \\ &\text{subject to} \quad \mathbf{W}\geq\mathbf{0} \text{ and } \mathbf{H}\geq\mathbf{0}. \end{aligned} \tag{2}$$

Here $\|\cdot\|_F$ denotes the Frobenius norm of a matrix. However, the optimization problem in Eq. (2) is nonconvex with respect to both the factors $\mathbf{W}$ and $\mathbf{H}$. To resolve this, most NMF algorithms divide the problem into two simpler subproblems which have closed-form solutions. The convex subproblem is solved by keeping one factor fixed while updating the other, alternating and iterating until convergence. One of the most popular techniques to minimize the subproblems is the method of multiplicative updates (MU) proposed by Lee and Seung [23]. This procedure is essentially a rescaled version of gradient descent. Its simple implementation comes at the expense of a much slower convergence.

More appealing are alternating least squares methods and their variants. Among these, HALS proves to be highly efficient [6]. Without being exhaustive, we would also like to point out the interesting work by Gillis and Glineur [15] as well as by Kim et al. [18] who proposed improved and accelerated HALS algorithms for computing NMF.

2.2.2. Separable schemes

Another approach to compute NMF is based on the idea of column subset selection. In this method, k columns of the input matrix are chosen to form the factor matrix $\mathbf{W} := \mathbf{X}(:, J)$, where J denotes the index set. The factor matrix $\mathbf{H}$ is found by solving the following optimization problem:

$$\text{minimize} \quad f(\mathbf{H}) = \|\mathbf{X}-\mathbf{W}(:,J)\mathbf{H}\|_F^2 \quad \text{s.t.} \quad \mathbf{H}\geq\mathbf{0}. \tag{3}$$

In context of NMF, this approach is appealing if the input matrix is separable [1]. This means it must be possible to select basis vectors for $\mathbf{W}$ from the columns of the input matrix $\mathbf{X}$. In this case, selecting actual columns from $\mathbf{X}$ preserves the underlying structure of the data and allows a meaningful interpretation. This assumption is intrinsic in many applications, e.g., document classification and blind hyperspectral unmixing [14]. However, this approach has limited potential in applications where the data is dense or noisy.

These separable schemes are not unique and can be obtained through various algorithms. Finding a meaningful column subset is explored in the CX decomposition [4], which extracts columns that best describe the data. In addition, the CUR decomposition [26] leverages statistical significance of both columns and rows to improve interpretability, leading to near-optimal decompositions. Another interesting algorithm to compute the near-separable NMF was proposed by [39]. This algorithm finds conical hulls in which smaller subproblems can be computed in parallel in 1D or 2D. For details on ongoing research in column selection algorithms, we refer the reader to [5,36,37].

2.3. Probabilistic framework

In the era of 'big data', probabilistic methods have become indispensable for computing low-rank matrix approximations. The central concept is to utilize randomness in order to form a surrogate matrix which captures the essential information of a high-dimensional input matrix. This assumes that the input matrix features low-rank structure, i.e., the effective rank is smaller than its ambient dimensions. Following Halko et al. [17], the probabilistic framework for low-rank approximations proceeds as follows.

Let $\mathbf{A}$ be an $m \times n$ matrix, without loss of generality we assume that $n \leq m$. First, we aim to approximate the range of $\mathbf{A}$. While the SVD provides the best possible basis in a least-square sense, a near-optimal basis can be obtained using random projections

$$\mathbf{Y} := \mathbf{A}\boldsymbol{\Omega}, \tag{4}$$

where $\boldsymbol{\Omega} \in \mathbb{R}^{n\times k}$ is a random test matrix. Recall, that the target rank of the approximation is denoted by the integer k, and is assumed to be $k \ll n$. Typically, the entries of $\boldsymbol{\Omega}$ are independently and identically drawn from the standard normal distribution. Next, the QR-decomposition of $\mathbf{Y}$ is used to form a matrix $\mathbf{Q} \in \mathbb{R}^{m\times k}$ with orthogonal columns. Thus, this matrix forms a near-optimal normal basis for the input matrix such that

$$\mathbf{A} \approx \mathbf{Q}\mathbf{Q}^\top\mathbf{A} \tag{5}$$

is satisfied. Finally, a smaller matrix $\mathbf{B} \in \mathbb{R}^{k\times n}$ is computed by projecting the input matrix to low-dimensional space

$$\mathbf{B} := \mathbf{Q}^\top\mathbf{A}. \tag{6}$$

Hence, the input matrix can be approximately decomposed as

$$\mathbf{A} \approx \mathbf{QB}. \tag{7}$$

This process preserves the geometric structure in an Euclidean sense. The smaller matrix $\mathbf{B}$ is, in many applications, sufficient to construct a desired low-rank approximation. The approximation quality can be controlled by oversampling and the concept of power iterations (for more details see Section 3).

Martinsson [27] provides the following simplified description of the expected error of the outlined procedure:

$$\mathbb{E}\,\|\mathbf{A}-\mathbf{QB}\|_2 \leq \left[1+\sqrt{\frac{k}{p-1}}+\frac{e\sqrt{k+p}}{p}\cdot\sqrt{n-k}\right]^{\frac{1}{2q+1}}\sigma_{k+1}(\mathbf{A}).$$

Here, p denotes the oversampling parameter and q the number of additional power iterations. It follows that as p increases, the error tends towards the best possible approximation error, i.e., the singular value $\sigma_{k+1}(\mathbf{A})$. A rigorous error analysis is provided by [17].

We refer the reader to the surveys by Halko et al. [17], Mahoney [25], Drineas and Mahoney [8] and Martinsson [27] for more detailed discussions of randomized algorithms. For implementations of the randomized SVD and related low-rank approximations see [11,31,35].

3. Randomized nonnegative matrix factorization

High-dimensional data pose a computational challenge for deterministic nonnegative matrix factorization, despite modern optimization techniques. Indeed, the costs of solving the optimization problem formulated in Eq. (2) can be prohibitive. Our motivation is to use randomness as a strategy to ease the computational demands of extracting low-rank features from high-dimensional data. Specifically, a randomized hierarchical alternating least squares (HALS) algorithm is formulated to efficiently compute the nonnegative matrix factorization.

3.1. Hierarchal alternating least squares

Block coordinate descent (BCD) methods are a universal approach to algorithmic optimization [38]. These iterative methods fix a block of components and optimize with respect to the remaining components. Following this philosophy, the HALS algorithm unbundles the original problem into a sequence of simpler optimization problems. This allows the efficient computation of the NMF [6].

Suppose that we update $\mathbf{W}$ and $\mathbf{H}$ by fixing most terms except for the block comprised of the jth column $\mathbf{W}_{(:,j)}$ and jth row $\mathbf{H}_{(j,:)}$. Thus, each subproblem is essentially reduced to a smaller minimization. HALS approximately minimizes the cost function in Eq. (2) with respect to the remaining $k-1$ components

$$\text{minimize} \quad J_j(\mathbf{W}_{(:,j)},\mathbf{H}_{(j,:)}) = \|\mathbf{X}^{(j)}-\mathbf{W}_{(:,j)}\mathbf{H}_{(j,:)}\|_F^2, \tag{8}$$

where $\mathbf{X}^{(j)}$ is the jth residual

$$\mathbf{X}^{(j)} := \sum_{i\neq j}^{k}\mathbf{W}_{(:,i)}\mathbf{H}_{(i,:)}. \tag{9}$$

This can be viewed as a decomposition of the residual [19]. Then, it is simple to derive the gradients to find the stationary points for both components

$$0 = \frac{\partial J_j}{\partial \mathbf{W}_{(:,j)}} = \mathbf{W}_{(:,j)}\mathbf{H}_{(j,:)}\mathbf{H}_{(j,:)}^\top - \mathbf{X}^{(j)}\mathbf{H}_{(j,:)}^\top, \tag{10}$$

$$0 = \frac{\partial J_j}{\partial \mathbf{H}_{(j,:)}} = \mathbf{H}_{(j,:)}^\top\mathbf{W}_{(:,j)}^\top\mathbf{W}_{(:,j)} - \mathbf{X}^{(j)\top}\mathbf{W}_{(:,j)}. \tag{11}$$

Hence, the update rules for the jth component of $\mathbf{W}$ and $\mathbf{H}$ are

$$\mathbf{W}_{(:,j)} \leftarrow \frac{1}{\mathbf{H}_{(j,:)}\mathbf{H}_{(j,:)}^\top}[\mathbf{X}^{(j)}\mathbf{H}_{(j,:)}^\top]_+, \tag{12}$$

$$\mathbf{H}_{(j,:)} \leftarrow \frac{1}{\mathbf{W}_{(:,j)}^\top\mathbf{W}_{(:,j)}}[\mathbf{X}^{(j)\top}\mathbf{W}_{(:,j)}]_+. \tag{13}$$

The maximum operator ensures that the components remain nonzero, defined as $[x]_+ := \max(0,x)$.

For an efficient implementation the update rules can be simplified [18] to the following:

$$\mathbf{W}_{(:,j)} \leftarrow \left[\mathbf{W}_{(:,j)} + \frac{\left[\mathbf{XH}^\top\right]_{(:,j)} - \left[\mathbf{WHH}^\top\right]_{(:,j)}}{[\mathbf{HH}^\top]_{(j,j)}}\right]_+, \tag{14}$$

$$\mathbf{H}_{(j,:)} \leftarrow \left[\mathbf{H}_{(j,:)} + \frac{\left[\mathbf{X}^\top\mathbf{W}\right]_{(:,j)} + \left[\mathbf{H}^\top\mathbf{W}^\top\mathbf{W}\right]_{(:,j)}}{[\mathbf{W}^\top\mathbf{W}]_{(j,j)}}\right]_+. \tag{15}$$

3.2. Randomized hierarchal alternating least squares

Employing randomness, we reformulate the optimization problem in Eq. (2) as a low-dimensional optimization problem. Specifically, the high-dimensional $m\times n$ input matrix $\mathbf{X}$ is replaced by the $k\times n$ surrogate matrix $\mathbf{B}$, which is formed as described in Section 2.3. Thus we yield

$$\begin{aligned}&\text{minimize} \quad \tilde{f}(\tilde{\mathbf{W}},\mathbf{H}) = \|\mathbf{B}-\tilde{\mathbf{W}}\mathbf{H}\|_F^2\\ &\text{subject to} \quad \mathbf{Q}\tilde{\mathbf{W}} \geq \mathbf{0} \text{ and } \mathbf{H} \geq \mathbf{0}.\end{aligned} \tag{16}$$

The nonnegativity constraints need to apply to the high-dimensional factor matrix $\mathbf{W}$, but not necessarily to $\tilde{\mathbf{W}}$. The matrix $\tilde{\mathbf{W}}$ can be rotated back to high-dimensional space using the following approximate relationship $\mathbf{W}\approx\mathbf{Q}\tilde{\mathbf{W}}$. Thus, Eq. (16) can only be solved approximately, since $\mathbf{QQ}^\top \neq \mathbf{I}$. Further, there is no reason that $\tilde{\mathbf{W}}$ has nonnegative entries, yet the low-dimensional projection will decrease the objective function in Eq. (16). A proof similar to [7] can be demonstrated. Now, we formulate the randomized HALS algorithm as

$$\text{minimize} \quad \tilde{J}_j(\tilde{\mathbf{W}}_{(:,j)},\mathbf{H}_{(j,:)}) = \|\mathbf{B}^{(j)}-\tilde{\mathbf{W}}_{(:,j)}\mathbf{H}_{(j,:)}\|_F^2, \tag{17}$$

where $\mathbf{B}^{(j)}$ is the jth compressed residual

$$\mathbf{B}^{(j)} := \sum_{i\neq j}^{k}\tilde{\mathbf{W}}_{(:,i)}\mathbf{H}_{(i,:)}. \tag{18}$$

The update rules for the components are as follows

$$\mathbf{H}_{(j,:)} \leftarrow \left[\mathbf{H}_{(j,:)} + \frac{\left[\mathbf{B}^\top\tilde{\mathbf{W}}\right]_{(:,j)} + \left[\mathbf{H}^\top\tilde{\mathbf{W}}^\top\tilde{\mathbf{W}}\right]_{(:,j)}}{[\tilde{\mathbf{W}}^\top\tilde{\mathbf{W}}]_{(j,j)}}\right]_+, \tag{19}$$

$$\tilde{\mathbf{W}}_{(:,j)} \leftarrow \tilde{\mathbf{W}}_{(:,j)} + \frac{\left[\mathbf{BH}^\top\right]_{(:,j)} - \left[\tilde{\mathbf{W}}\mathbf{HH}^\top\right]_{(:,j)}}{[\mathbf{HH}^\top]_{(j,j)}}. \tag{20}$$

Then, we employ the following scheme to update the jth component in high-dimensional space, followed by projecting the updated factor matrix $\mathbf{W}$ back to low-dimensional space

$$\tilde{\mathbf{W}}_{(:,j)} \leftarrow \mathbf{Q}^\top[\mathbf{Q}\tilde{\mathbf{W}}_{(:,j)}]_+. \tag{21}$$

The computational steps are summarized in Algorithm 1.

Remark 1 (Random test matrix)**.** The entries $\boldsymbol{\omega}_{ij}$ of the random test matrix $\boldsymbol{\Omega}$ are drawn independently from the uniform distribution in the interval of [0, 1]. Nonnegative random entries perform better than Gaussian distributed entries, and seem to be the natural choice in the context of nonnegative data.

Algorithm 1. Randomized HALS for NMF: $\mathbf{X} \approx \mathbf{WH}$.

Require: A nonnegative matrix $\mathbf{X}$ of dimension $m \times n$, and target rank k.

Optional: Parameters p and q for oversampling, and power iterations.

$l = k + p$ — Slight oversampling

$\mathbf{\Omega} = \texttt{rand}(n, l)$ — Generate sampling matrix $\mathbf{\Omega} \in \mathbb{R}^{n \times l}$

$\mathbf{Y} = \mathbf{X\Omega}$ — Form basis $\mathbf{Y} \in \mathbb{R}^{m \times l}$

for $j = 1, \ldots, q$ — Optional: subspace iterations

  $[\mathbf{Q}, \sim] = \texttt{qr}(\mathbf{Y})$

  $[\mathbf{Q}, \sim] = \texttt{qr}(\mathbf{X}^\top \mathbf{Q})$

  $\mathbf{Y} = \mathbf{XQ}$

$[\mathbf{Q}, \sim] = \texttt{qr}(\mathbf{Y})$ — Form orthonormal basis $\mathbf{Q} \in \mathbb{R}^{m \times l}$

$\mathbf{B} = \mathbf{Q}^\top \mathbf{X}$ — Form smaller matrix $\mathbf{B} \in \mathbb{R}^{l \times n}$

Initialize nonnegative factors $\mathbf{W} \in \mathbb{R}^{m \times k}$, $\tilde{\mathbf{W}} \in \mathbb{R}^{l \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times n}$

repeat

$\mathbf{R} = \mathbf{B}^\top \tilde{\mathbf{W}}$ — $\mathbf{R} \in \mathbb{R}^{n \times k}$

$\mathbf{S} = \tilde{\mathbf{W}}^\top \tilde{\mathbf{W}}$ — $\mathbf{S} \in \mathbb{R}^{k \times k}$

for $j = 1, \ldots, k$ — Update $\mathbf{H}$ row by row

  $\mathbf{H}(j,:) = \mathbf{H}(j,:) + (\mathbf{R}(:,j) - \mathbf{H}^\top \mathbf{S}(:,j))/\mathbf{S}(j,j)$

  $\mathbf{H}(j,:) = \mathbf{max}(0, \mathbf{H}(j,:))$ — Elementwise maximum operator

$\mathbf{T} = \mathbf{BH}^\top$ — $\mathbf{T} \in \mathbb{R}^{k \times k}$

$\mathbf{V} = \mathbf{HH}^\top$ — $\mathbf{V} \in \mathbb{R}^{k \times k}$

for $j = 1, \ldots, k$ — Update $\mathbf{W}$ column by column

  $\tilde{\mathbf{W}}(:,j) = \tilde{\mathbf{W}}(:,j) + (\mathbf{T}(:,j) - \tilde{\mathbf{W}}\mathbf{V}(:,j))/\mathbf{V}(j,j)$

  $\mathbf{W}(:,j) = \mathbf{max}(0, \mathbf{QW}(:,j))$ — Elementwise maximum operator

  $\tilde{\mathbf{W}}(:,j) = \mathbf{Q}^\top \mathbf{W}(:,j)$ — Rotate to low-dimensional space

if stopping criterion or maximum number of iterations is reached

  Return: Nonnegative factor matrices $\mathbf{W} \in \mathbb{R}^{m \times k}$ and $\mathbf{H} \in \mathbb{R}^{k \times n}$

Remark 2 (Oversampling)**.** Oversampling is required to find a good basis matrix [17]. This is because real-world data often do not have exact rank. Thus, instead of just computing k random projections we compute $k + p$ in order to form the basis matrix $\mathbf{Y}$. Specifically, this procedure increases the probability that $\mathbf{Y}$ approximately captures the column space of $\mathbf{X}$. Our experiments show that small oversampling values of about $p = \{10, 20\}$ achieve good approximation results.

Remark 3 (Power scheme)**.** Power iterations can further help to improve the quality of the basis matrix [30]. The idea is to preprocess the input matrix in order to sample from a matrix which has a faster decaying singular value spectrum. Therefore, Eq. (4) is replaced by

$$\mathbf{Y} = \left((\mathbf{XX}^*)^q \mathbf{X}\right)\mathbf{\Omega} \tag{22}$$

where q specifies the number of power iterations. The drawback, however, is that additional passes over the input matrix are required. Note that the direct implementation of power iteration is numerically unstable. Thus, subspace iterations should be used instead [16,27].

Remark 4 (Initialization)**.** The performance of NMF algorithms depends on the procedure used for initializing the factor matrices. We refer to [21] for an excellent discussion on this topic. We proceed by simply initializing the factor matrices with Gaussian entries, where negative elements are set to 0.

Remark 5 (Stopping criterion)**.** Predefining a maximum number of iterations is not satisfactory in practice. Instead, we aim to stop the algorithm, if a suitable measure has reached some convergence tolerance. For a discussion on stopping criteria, for instance, see [15]. The algorithm is terminated if the convergence rate is smaller than the stopping condition ϵ or if the following quantity is below the stopping criteria

$$\log_{10}\left[||\mathbf{B} - \tilde{\mathbf{W}}\mathbf{H}||_F\right]. \tag{23}$$

Table 1
Summary of the computational results for the MNIST dataset. The target rank is $k = 16$. Deterministic HALS is the baseline to compute speedups.

	Time (s)	Speedup	Iterations	Error
Deterministic HALS	6.27	–	100	0.543
Randomized HALS	2.38	2.6	100	0.549
Compressed MU	3.41	1.8	185	0.560
SVD	2.86	2.2	–	0.494

4. Experimental evaluation

In the following, we evaluate the proposed randomized algorithm and compare it to the deterministic HALS algorithm as implemented in the *scikit-learn* package [29]. Througout all experiments, we set the oversampling parameter to $p = 20$ and the number of subspace iterations to $q = 2$ for the randomized algorithm. Further, we set the tolerance parameter of the stopping condition to 1e−4 and limit the number of iterations to a maximum of 100. For comparison, we also provide the results of the compressed MU algorithm using the default settings as proposed by Tepper and Sapiro [32].[1] All computations are performed on a laptop with Intel Core i7-7700K CPU Quad-Core 4.20 GHz, 64GB fast memory.

4.1. Handwritten digits

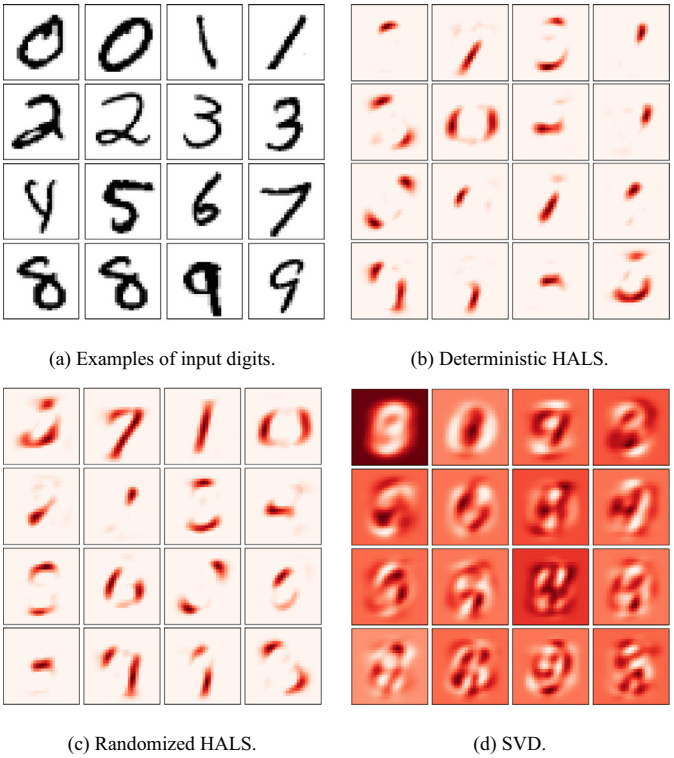
In practice, it is often of great interest to extract features from data in order to characterize the underlying structure or to preprocess the data. The extracted features can then be used, for instance, to visualize the data or for classification.

In the following we use the MNIST (Modified National Institute of Standards and Technology) database of handwritten digits, which comprises 60,000 training and 10,000 testing images, for demonstration (the data are obtained from http://yann.lecun.com/exdb/mnist/). Each image patch is of dimension 28×28 and represents a digit between 0 and 9. Fig. 1 shows the first 16 dominant basis images computed via the deterministic and randomized HALS as well as the SVD. Compared to the holistic features found by the SVD, NMF computes a parts-based representation of the digits. Hence, each digit can be formed as an additive combination of the individual parts. Furthermore, the basis images are simple to interpret.

Table 1 summarizes the computational results and shows that the randomized algorithms achieve a near-optimal reconstruction error, while significantly reducing computation. Specifically, the costs per iteration of the randomized HALS algorithms are substantially lower than of the deterministic algorithm.

Next, we investigate the question of whether the quality of the features computed by the randomized algorithm are sufficient for classification. We use the basis images to first project the data into low-dimensional space, then we use the k-nearest-neighbors method, with $k = 3$, for classification. The results for both the training and test samples are shown in Table 2.

[1] Note, that Tepper and Sapiro [32] have also used the active set and the alternating direction method of multipliers for computing the NMF. While both of these methods perform better than the MU algorithm in several empirical experiments, we faced some numerical issues in applications of interest to us. Hence, we present only the results of the compressed MU algorithm in the following. Further, Tepper and Sapiro [32] have used a vanilla flavored implementation of the power scheme; this tends to distort the basis due to round-off errors [27]. In practice it is recommended to implement the power scheme via the concept of subspace iterations.



(a) Examples of input digits. (b) Deterministic HALS. (c) Randomized HALS. (d) SVD.

Fig. 1. Dominant basis images extracted from the MNIST dataset (a) using the deterministic (b) and randomized HALS (c) algorithm as well as the SVD (d). Unlike the SVD, the NMF provides parts-based (sparse) features characterizing the underlying structure of the data. Note that the data are not mean centered.

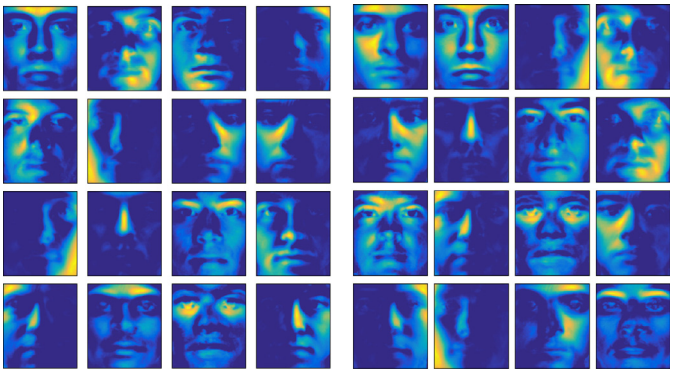
Table 2
Classification results of the MNIST dataset using the k-nearest-neighbors method. The results show that the overall predictive accuracy of both the randomized and deterministic features is similar.

	Precision	Recall	F1-score
(a) Training data.			
Deterministic HALS	0.97	0.97	0.97
Randomized HALS	0.97	0.97	0.97
Compressed MU	0.97	0.97	0.97
SVD	0.98	0.98	0.98
(b) Test data.			
Deterministic HALS	0.94	0.94	0.94
Randomized HALS	0.94	0.94	0.94
Compressed MU	0.94	0.94	0.94
SVD	0.96	0.96	0.96

The reconstruction error of the randomized and compressed algorithm are slightly poorer than that of the deterministic HALS algorithm. Interestingly, a similar classification performance in terms of precision, recall, and the F1-score is achieved. Note, that it is not the aim to design the best possible classifier, but to compare the randomized to the deterministic algorithms. We refer the reader to the work by Leucan et al. [22] for a comprehensive discussion of digit recognition techniques.

4.2. Facial feature extraction

Feature extraction from facial images is an essential preprocessing step for facial recognition. An ordinary approach is to use the SVD or PCA for this task, which was first studied by Kirby and Sirovich [20] and later by Turk and Pentland [33]. The resulting basis images represent 'shadows' of the faces, the so-called eigenfaces. However, instead of learning holistic representations, it seems more natural to learn a parts-based representation of the facial data. This was first demonstrated in the seminal work by Lee and Seung [23] using the NMF. The corresponding basis images are more robust to occlusions and are easier to interpret.



(a) Deterministic HALS. (b) Randomized HALS.

Fig. 2. Dominant basis images encoding the facial features. The proposed randomized NMF algorithm faithfully encodes the facial features.

Table 3
Summary of the computational results for the Yale face database B. The randomized HALS algorithm achieves a substantial speedup by a factor of 13.4. Baseline for speedup is deterministic HALS. The target rank is $k = 16$.

	Time (s)	Speedup	Iterations	Error
Deterministic HALS	21.23	–	100	0.239
Randomized HALS	1.58	13.4	100	0.242
Compressed MU	5.10	4.1	380	0.245

In the following we use the downsampled cropped Yale face database B [12]. The dataset comprises 2410 grayscale images, cropped and aligned. Each image is of dimension 192×168. Thus, we yield a data matrix of dimension 32, 256×2410 after vectorizing and stacking the images. Fig. 2 shows the 16 dominant features extracted via the deterministic and randomized HALS, which characterize some facial features. The randomized algorithm achieves a substantial speed-up of a factor of about 13.4, while the reconstruction error remains near-optimal, as shown in Table 3. In comparison, the compressed MU algorithm performs slightly poorer overall.

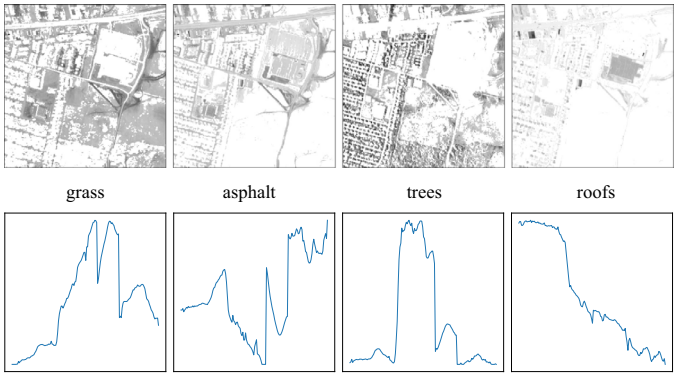
4.3. Hyperspectral unmixing

Hyperspectral imaging is often used in order to determine what materials and underlying processes are present in a scene. It uses data collected from across the electromagnetic spectrum. The difficulty, however, is that pixels of hyperspectral images commonly consist of a mixture of several materials. Thus, the aim of hyperspectral unmixing (HU) is to separate the pixel spectra into a collection of the so-called endmembers and abundances. Specifically, the endmembers represent the pure materials, while the abundances reflect the fraction of each endmember that is present in the pixel [3]. The NMF represents a simple linear mixing model for blind HU in form of

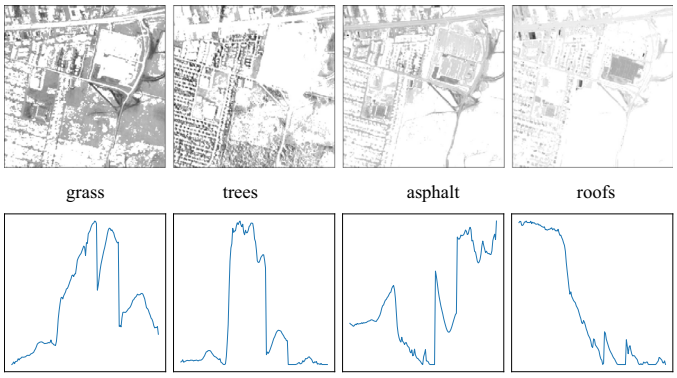
$$p_i = \mathbf{W}\mathbf{H}_{(:,i)} \tag{24}$$

where the ith pixel is denoted as p_i. The basis matrix $\mathbf{W}$ represents the spectral signatures of the endmembers, while the ith column of the weight matrix $\mathbf{H}$ represents the abundances of the corresponding pixel.

In the following we use the popular 'urban' hyperspectral image for demonstration (the data are obtained from http://www.agc.army.mil/). The hyperspectral image is of dimension 307×307 pixels, each corresponding to an area of 2×2 meters. Further, the im-



(a) Deterministic HALS.



(b) Randomized HALS.

Fig. 3. Dominant basis images (endmembers) and abundance maps extracted from the 'urban' hyperspectral image. The extracted spectral signatures by both the randomized and deterministic algorithms are very similar, while the abundance maps show slight differences, specifically in peaks in the spectra.

Table 4
Summary of the computational results for the hyperspectral image. Randomized HALS achieves a speedup of about 2.2, while attaining the same relative error as the deterministic algorithm. The target rank is $k = 4$. Baseline for speedup is deterministic HALS.

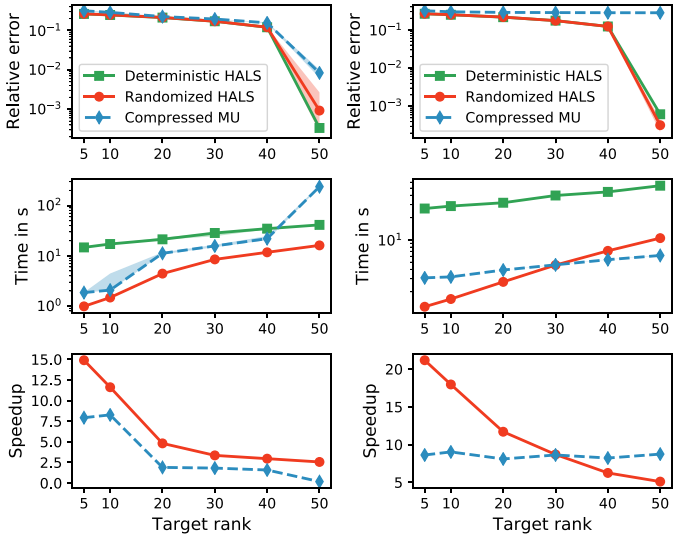
	Time (s)	Speedup	Iterations	Error
Deterministic HALS	1.34	–	100	0.040
Randomized HALS	0.6	2.2	100	0.040
Compressed MU	8.78	0.15	550	0.042

age consists of 210 spectral bands; however, we omit several channels due to dense water vapor and atmospheric effects. Thus, we use only 162 bands for the following analysis, which aims to automatically extract the four endmembers: Asphalt, grass, tree and roof. Fig. 3 shows the $k = 4$ dominant basis images reflecting the four endmembers as well as the corresponding abundance map. Both the deterministic and randomized algorithms faithfully reflect the four endmembers.

Table 4 quantifies the results. The randomized HALS and compressed MU algorithms require more iterations to converge compared to the deterministic HALS. Speedup of randomized HALS is considerable by a factor of about 2.2, whereas performance of compressed MU is weak for both speedup and error.

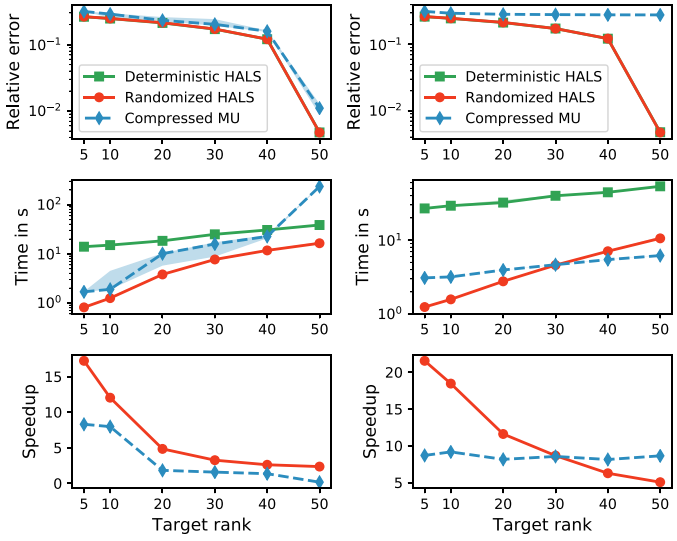
4.4. Computational performance

We use synthetic nonnegative data to contextualize the computational performance of the algorithms. Specifically, we construct low-rank matrices consisting of nonnegative elements drawn from the Gaussian distribution.



(a) Dimension: $50,000 \times 3,000$. (b) Dimension: $20,000 \times 15,000$.

Fig. 4. The randomized HALS algorithm outperforms on both tall-and-skinny (a), and fat (b) synthetic data matrices of rank 50. The compressed MU algorithm fails to converge in (b). Baseline for speedup is deterministic HALS.



(a) Dimension: $50,000 \times 3,000$. (b) Dimension: $20,000 \times 15,000$.

Fig. 5. Synthetic data matrices of rank 50, perturbed with 10% white noise.

First, we compute the NMF for low-rank matrices of dimension $50,000 \times 3000$ and $20,000 \times 15,000$, both of rank $r = 50$. Fig. 4 shows the relative error, the timings, and the speedup for varying target-ranks k, averaged over 20 runs. First, we notice that the randomized HALS algorithm shows a significant speedup over the deterministic HALS algorithm by a factor of 2.5 to 22, while achieving a high accuracy. Here, we have limited the maximum number of iterations to 100 for both the randomized and deterministic HALS algorithm. If required, an even higher accuracy could be achieved by allowing for a larger number of iterations.

The MU algorithm is known to require a larger number of iterations. Thus, we have allowed for a maximum number of iterations of 5,000. Despite the large number of iterations, the compressed MU algorithms shows a patchy performance in comparison. While the results for small target ranks are satisfactory, the algorithm has difficulties in approximating factors of larger ranks with high accuracy. Further, we see that the compressed MU algorithm does not converge on the $20,000 \times 15,000$ matrix.

Next, the evaluation is repeated in presence of 10% additive nonnegative Gaussian noise. The results are shown in Fig. 5 and show a similar picture. As before, the randomized algorithm outperforms the deterministic, while showing a near-optimal relative error. The compressed MU algorithm shows the same performance issues as above.

5. Conclusion

Massive nonnegative data poses a computational challenge for deterministic NMF algorithms. However, randomized algorithms are a promising alternative for computing the approximate nonnegative factorization. The computational complexity of the proposed randomized algorithm scales with the target rank rather than ambient dimension of the measurement space. Hence, the computational advantage becomes pronounced with increasing dimensions of the input matrix.

The randomized HALS algorithm substantially reduces the computational demands, while achieving near-optimal reconstruction results. Thereby, the trade-off between precision and speed can be controlled via the concept of oversampling and the number of power iterations. We prose $p = 20$ and the computation of $q = 2$ subspace iterations as default values. These settings show a good performance throughout all our experiments. Overall, the randomized HALS algorithm shows considerable advantages over the deterministic HALS and the previously proposed compressed MU algorithm.

Future research will investigate a GPU-accelerated implementation of the proposed randomized HALS algorithm. Furthermore, it is an exciting research direction to extend the presented ideas to nonnegative tensor factorization using the randomized framework proposed by Erichson et al. [10].

Acknowledgments

We would like to express our gratitude to the two anonymous reviewers for their helpful feedback which allowed us improve the manuscript. NBE and JNK acknowledge support from an SBIR grant through SURVICE Inc. (FA9550-17-C-0007).

References

[1] S. Arora, R. Ge, R. Kannan, A. Moitra, Computing a nonnegative matrix factorization–provably, in: Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, ACM, 2012, pp. 145–162.
[2] M.W. Berry, M. Browne, A.N. Langville, V.P. Pauca, R.J. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization, Comput. Stat. Data Anal. 52 (1) (2007) 155–173.
[3] J.M. Bioucas-Dias, A. Plaza, N. Dobigeon, M. Parente, Q. Du, P. Gader, J. Chanussot, Hyperspectral unmixing overview: geometrical, statistical, and sparse regression-based approaches, IEEE J. Sel. Top. Appl. Earth Obs. Remote Sens. 5 (2) (2012) 354–379.
[4] C. Boutsidis, P. Drineas, M. Magdon-Ismail, Near-optimal column-based matrix reconstruction, SIAM J. Comput. 43 (2) (2014) 687–717.
[5] C. Boutsidis, D.P. Woodruff, Optimal cur matrix decompositions, SIAM J. Comput. 46 (2) (2017) 543–589.
[6] A. Cichocki, P. Anh-Huy, Fast local algorithms for large scale nonnegative matrix and tensor factorizations, IEICE Trans. Fundam. Electr. Commun. Comput. Sci. 92 (3) (2009) 708–721.
[7] J. Cohen, R.C. Farias, P. Comon, Fast decomposition of large nonnegative tensors, IEEE Signal Process. Lett. 22 (7) (2015) 862–866.
[8] P. Drineas, M.W. Mahoney, Randnla: randomized numerical linear algebra, Commun. ACM 59 (6) (2016) 80–90, doi:10.1145/2842602.
[9] C. Eckart, G. Young, The approximation of one matrix by another of lower rank, Psychometrika 1 (3) (1936) 211–218.
[10] N.B. Erichson, K. Manohar, S.L. Brunton, J.N. Kutz, Randomized CP tensor decomposition, Preprint arXiv:1703.09074 (2017) 1–29.
[11] N.B. Erichson, S. Voronin, S.L. Brunton, J.N. Kutz, Randomized matrix decompositions using r, arXiv preprint arXiv:1608.02148 (2016).
[12] A.S. Georghiades, P.N. Belhumeur, D.J. Kriegman, From few to many: illumination cone models for face recognition under variable lighting and pose, IEEE Trans. Pattern Anal. Mach. Intell. 23 (6) (2001) 643–660.
[13] N. Gillis, The Why and How of Nonnegative Matrix Factorization, in: Regularization, Optimization, Kernels, and Support Vector Machines, Chapman and Hall/CRC, 2014, pp. 257–291.
[14] N. Gillis, Introduction to nonnegative matrix factorization, arXiv preprint arXiv:1703.00663 (2017).
[15] N. Gillis, F. Glineur, Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization, Neural Comput. 24 (4) (2012) 1085–1105.
[16] M. Gu, Subspace iteration randomization and singular value problems, SIAM J. Sci. Comput. 37 (3) (2015) 1139–1173.
[17] N. Halko, P.-G. Martinsson, J.A. Tropp, Finding structure with randomness: probabilistic algorithms for constructing approximate matrix decompositions, SIAM Rev. 53 (2) (2011) 217–288, doi:10.1137/090771806.
[18] J. Kim, Y. He, H. Park, Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework, J. Global Optim. 58 (2) (2014) 285–319.
[19] K. Kimura, Y. Tanaka, M. Kudo, A Fast Hierarchical Alternating Least Squares Algorithm for Orthogonal Nonnegative Matrix Factorization, in: Proceedings of the Sixth Asian Conference on Machine Learning, in: Proceedings of Machine Learning Research, PMLR, 39, 2015, pp. 129–141.
[20] M. Kirby, L. Sirovich, Application of the Karhunen–Loeve procedure for the characterization of human faces, Pattern Anal. Mach. Intell. IEEE Trans. 12 (1) (1990) 103–108.
[21] A.N. Langville, C.D. Meyer, R. Albright, J. Cox, D. Duling, Algorithms, initializations, and convergence for the nonnegative matrix factorization, arXiv preprint arXiv:1407.7299 (2014).
[22] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[23] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791, doi:10.1038/44565.
[24] D.D. Lee, H.S. Seung, Algorithms for Non-negative Matrix Factorization, in: Advances in Neural Information Processing Systems, 2001, pp. 556–562.
[25] M.W. Mahoney, Randomized algorithms for matrices and data, Found. Trends Mach. Learn. 3 (2) (2011) 123–224, doi:10.1561/2200000035.
[26] M.W. Mahoney, P. Drineas, Cur matrix decompositions for improved data analysis, Proc. Natl. Acad. Sci. 106 (3) (2009) 697–702.
[27] P.-G. Martinsson, Randomized methods for matrix computations and analysis of high dimensional data, Preprint arXiv:1607.01649 (2016) 1–55.
[28] P. Paatero, U. Tapper, Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values, Environmetrics 5 (2) (1994) 111–126.
[29] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: machine learning in python, J. Mach. Learn. Res. 12 (2011) 2825–2830.
[30] V. Rokhlin, A. Szlam, M. Tygert, A randomized algorithm for principal component analysis, SIAM J. Matrix Anal. Appl. 31 (3) (2009) 1100–1124.
[31] A. Szlam, Y. Kluger, M. Tygert, An implementation of a randomized algorithm for principal component analysis, Preprint arXiv:1412.3510 (2014) 1–13.
[32] M. Tepper, G. Sapiro, Compressed nonnegative matrix factorization is fast and accurate, IEEE Trans. Signal Process. 64 (9) (2016) 2269–2283.
[33] M.A. Turk, A.P. Pentland, Face recognition using eigenfaces, in: Proceedings on Computer Vision and Pattern Recognition, IEEE, 1991, pp. 586–591.
[34] M. Udell, A. Townsend, Nice latent variable models have log-rank, arXiv preprint arXiv:1705.07474 (2017).
[35] S. Voronin, P.-G. Martinsson, Rsvdpack: subroutines for computing partial singular value decompositions via randomized sampling on single core, multi core, and GPU architectures, Preprint arXiv:1502.05366 (2015) 1–15.
[36] S. Wang, Z. Zhang, Improving cur matrix decomposition and the nyström approximation via adaptive sampling, J. Mach. Learn. Res. 14 (1) (2013) 2729–2769.
[37] S. Wang, Z. Zhang, T. Zhang, Towards more efficient spsd matrix approximation and cur matrix decomposition, J. Mach. Learn. Res. 17 (210) (2016) 1–49.
[38] S.J. Wright, Coordinate descent algorithms, Math. Program. 151 (1) (2015) 3–34.
[39] T. Zhou, W. Bian, D. Tao, Divide-and-conquer anchoring for near-separable nonnegative matrix factorization and completion in high dimensions, in: Data Mining (ICDM), 2013 IEEE 13th International Conference on, IEEE, 2013, pp. 917–926.
[40] T. Zhou, D. Tao, Bilateral random projections, in: Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on, IEEE, 2012, pp. 1286–1290.