J Glob Optim
DOI 10.1007/s10898-016-0471-z

Accelerated parallel and distributed algorithm using limited internal memory for nonnegative matrix factorization

Duy Khuong Nguyen[1,2] · **Tu Bao Ho**[1,3]

Received: 19 June 2015 / Accepted: 23 September 2016

Abstract Nonnegative matrix factorization (NMF) is a powerful technique for dimension reduction, extracting latent factors and learning part-based representation. For large datasets, NMF performance depends on some major issues such as fast algorithms, fully parallel distributed feasibility and limited internal memory. This research designs a fast fully parallel and distributed algorithm using limited internal memory to reach high NMF performance for large datasets. Specially, we propose a flexible accelerated algorithm for NMF with all its L_1 L_2 regularized variants based on full decomposition, which is a combination of exact line search, greedy coordinate descent, and accelerated search. The proposed algorithm takes advantages of these algorithms to converges linearly at an over-bounded rate $(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}$ in optimizing each factor matrix when fixing the other factor one in the sub-space of passive variables, where r is the number of latent components, and μ and L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$. In addition, the algorithm can exploit the data sparseness to run on large datasets with limited internal memory of machines, which is is advanced compared to fast block coordinate descent methods and accelerated methods. Our experimental results are highly competitive with seven state-of-the-art methods about three significant aspects of convergence, optimality and average of the iteration numbers.

Keywords Non-negative matrix factorization · Accelerated anti-lopsided algorithm · Cooridinate descent algorithm · Parallel and distributed algorithm

✉ Duy Khuong Nguyen
khuongnd@gmail.com; khuongnd@jaist.ac.jp; khuongnd@vnu.edu.vn

[1] Japan Advanced Institute of Science and Technology, Nomi, Japan

[2] University of Engineering and Technology, Vietnam National University, Hanoi, Vietnam

[3] John von Neumann Institute, Vietnam National University, Ho Chi Minh City, Vietnam

Published online: 04 October 2016

1 Introduction

Nonnegative matrix factorization (NMF) is a powerful technique widely used in applications of data mining, signal processing, computer vision, bioinformatics, etc. [1]. Fundamentally, NMF has two main purposes. First, it reduces dimension of data making learning algorithms faster and more effective as they often work less effectively due to the curse of dimensionality [2]. Second, NMF helps extracting latent components and learning part-based representation, which are the significant distinction from other dimension reduction methods such as Principal Component Analysis (PCA), Independent Component Analysis (ICA), Vector Quantization (VQ), etc. This feature originates from transforming data into lower dimension of latent components and non-negativity constraints [3–5].

In the last decade of fast development, there were remarkable milestones. The two first milestones in early days of the NMF historical development were its mathematical formulations as positive matrix factorization with Byzantine algorithms [6] and as parts-based representation with a simple effective algorithm [5]. The last decade has witnessed the rapid NMF development [1,7]. Various works on NMF can be viewed in three major perspectives: variants of NMF, algorithms and applications. In particular, variants of NMF are based on either divergence functions [8,9], or constraints [10,11], or regularizations [12,13]. Most NMF algorithms were developed along two main directions: geometric greedy algorithms [14] and iterative multiplicative update algorithms. Although geometric greedy algorithms are usually fast, they are hard to trade off complexity, optimality, loss information and sparseness.

More recently, it is well recognized that the most challenging problems in iterative multiplicative update algorithms for NMF are fast learning, limited internal memory, parallel distributed computation, among others. In particular, fast learning is essential in learning NMF models from large datasets, and it is indeed difficult to carry out them when the number of variables is very large. In addition, the limited internal memory is one of the most challenging requirements for big data [15], because data has been exploring rapidly while the internal memory of nodes is always limited. Finally, parallel and distributed computation makes NMF applications feasible for big data [16].

To deal with these challenges, this work develops an accelerated algorithm for NMF and its L_1 L_2 regularized variants having several major advantages that are summarized in Table 1. In this paper, we contribute to four aspects as follows:

- *NMF and its variants* We fully decompose NMF and its L_1 L_2 regularized variants into non-negative quadratic programming problems. This decomposition makes the proposed algorithm flexible to adapt all L_1 L_2 regularized NMF in an unified framework that can trade-off the quality of information loss, sparsity and smoothness.
- *Algorithm* We employ an accelerated anti-lopsided algorithm for approximating solutions of non-negative quadratic programming. The algorithm reduces variable scaling problems to achieve a linear convergence rate $\mathcal{O}((1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r})$ in optimizing each factor matrix in the sub-space of passive variables, which is advanced to fast coordinate methods and accelerated methods in terms of efficiency as well as convergence rate. In addition, the size of the optimization problem is reduced into r ($r \ll m, n$), which is the smallest among the state-of-the-art methods. Furthermore, it is highly potential to be applied in alternating least squares methods for factorization models.
- *Parallel and distribution* The proposed algorithm is fully parallel and distributed on limited internal memory systems, which is crucial for big data when computing nodes having limited internal memory that cannot hold the whole dataset. Furthermore, the

Table 1 Comparison summary of NMF solvers

Criteria	Inexact		Exact				Accelerated			
	MUR	PrG	Qn	Nt	AcS	BlP	FCD	AcH	Ne	Alo
Guaranteed convergence	✗	✗	✗	✗	✗	✗	✗	✗	$\frac{1}{k^2}$	$[(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k$
Exploit data spareness	✗	✗	✗	✗	✗	✗	✓	✓	✗	✓
Required internal memory	$\mathcal{O}(mn+r(r+n+m))$									$\mathcal{O}(r(r+n))$
Fully parallel and distributed	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓
Optimization problem size	$r(m,n)$			r	$r(n,m)$			(m,n)	$r(m,n)$	r

✓ Means considered, and ✗ Means not considered
$\frac{1}{2} \le \mu \le L \le r$, $n \times m$ is the data matrix size, r is the number of latent components
$(m,n) = \max(m,n)$, and $r(m,n) = r.\max(m,n)$
MUR multiplicative update rule [5]; *PrG* projected gradient methods [17]; *Nt* newton-type methods [18]; *Qn* projected quasi-newton [19]; *AcS* fast active-set-like method [20]; *BlP* block principal pivoting method [21]; *FCD* fast coordinate descent methods with variable selection [22]; *AcH* accelerated hierarchical alternating least squares [23]; *Nev* Nesterov's optimal gradient method [24]; *Alo* the proposed method

proposed algorithm is convenient to implement for hybrid multi-core distributed systems because this algorithm works on each individual instance and each latent feature.

– *Comparison* This is the first time that state-of-the-art algorithms among different research directions for NMF are compared together.

2 Background and related works

2.1 Problem statement

Mathematically, NMF in Frobenius norm is defined as follows:

Definition 1 [*NMF*]: Given a dataset consisting of m vectors in n dimensions $V = [V_1, V_2, ..., V_m] \in \mathbb{R}_+^{n \times m}$, where each vector presents a data instance. NMF seeks to approximately factorize V into a product of two nonnegative factorizing matrices W^T and F, where $W \in \mathbb{R}_+^{r \times n}$ and $F \in \mathbb{R}_+^{r \times m}$ are coefficient matrix and latent component matrix, respectively, $V \approx W^T F$, in which the quality of the approximation can be guaranteed by the objective function in Frobenius norm as $J(V \| W^T F) = \| V - W^T F \|_2^2$.

Although NMF is a non-convex problem, optimizing each factor matrix when fixing the other one is a convex problem. In other words, F can be traced when W is fixed, and vice versa. Furthermore, F and W have different roles although they are symmetric in the objective function. W are latent components to represent data instances V by coefficients F. Hence, NMF can be considered as a latent factor model of latent components W, and learning this model is equivalent to find out latent components W. Therefore, in this chapter, we propose an accelerated parallel and distributed algorithm to learn NMF models W for large datasets.

2.2 Related works of nonnegative matrix factorization

NMF algorithms can be divided into two groups: greedy algorithms and iterative multiplicative update algorithms. The greedy algorithms [14] are often based on geometric

interpret-ability, and they can be extremely fast to deal with large datasets. However, it is hard to trade off complexity, optimality, loss information and sparseness. The iterative multiplicative update algorithms such as "two-block coordinate descent" often consist of two steps, each of them fixes one of two matrices to replace the other matrix for obtaining the convergence of the objective function. There are numerous studies on these algorithms, see Table 1, because NMF is nonconvex, though two steps corresponding to two non-negative least square (NNLS) sub-problems are convex [24,25]. In addition, various constraints and optimization strategies have been used to trade off the convexity, information loss, complexity, sparsity, and numerical instability.

Based on the optimization updating strategy, these iterative multiplicative update algorithms can be further divided into three sub-groups:

- *Inexact block coordinate descent* The algorithms' common characteristic is their usage of gradient methods to seek an approximate solution for NNLS problems, which is neither optimal nor fulfilling of fast approximations and accelerated conditions. Lee et al. [5] proposed the (basic) NMF problem and simple multiplicative updating rule (MUR) algorithm using a first-order gradient method to learn the part-based representation. Seung et al. [8] concerned rescaling gradient factors with a carefully selected learning rate to achieve faster convergence. Subsequently, Lin [26] modified MUR, which is theoretically proved getting a stationary point. However, that algorithm cannot improve the convergence rate. Berry et al. [27] projected nonnegative least square (PNLS) solutions into a nonnegative quadratic space by setting negative entries in the matrices to zero. Although this algorithm does not guarantee the convergence, it is widely applied in real applications. In addition, Bonettini et al. [28] used line search based on Amijo rule to obtain better solutions for matrices. Theoretically, this method can achieve optimal solutions for factor matrices as exact block coordinate descent group, but it slowly tends to stationary points because the line search is time-consuming.
- *Exact block coordinate descent* In contrast to the first sub-group, exact block coordinate descent methods obtains optimal solutions for two NNLS problems in each iteration. Zdunek et al. [19] employed a second-order quasi-Newton method with inverse of Hessian matrix to estimate the step size, aiming to a faster convergence than projected methods. However, this algorithm may be slow and non-stable because of the line search and the matrix inverse. Subsequently, Kim et al. [18] used rank-one to Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm to approximate the inverse of Hessian matrix. Furthermore, Lin [17] proposed several algorithms based on projected gradient methods and exact line search. Theoretically, this method can obtain more accurate solutions, however it is time-consuming because of exact line search, the large number of variables, and negative affects of nonnegative constraints. Moreover, Kim et al. [20,21] proposed two active-set methods based on Karush–Kuhn–Tucker (KKT) conditions, in which the variables are divided into two sets: a free set and an active set. Only the free set contains variables that can optimize the objective functions. Removing a number of redundant variables makes their algorithm improve the convergence rate significantly. However, the method still has heavy computation for large-scale problems.
- *Accelerated block coordinate descent* The accelerated methods use fast solution approximations satisfying accelerated conditions to reduce the complexity and to keep fast convergence. The accelerated conditions are different constraints in various methods to guarantee convergence to the optimal solution in comparison with the initial value. These accelerated methods are developed due to the limitation of inexact methods having slow convergence, and exact methods having high complexity. Particularly, the inexact meth-

ods have slow convergence because they requires a large number of iterations to converge to stationary points. Furthermore, the exact methods have high complexity in each iteration, however obtaining optimal solutions in every iteration is controversial because it can lead to zig-zag problems when optimizing a non-convex function of two independent sets of variables.

Firstly, Hsieh et al. [22] proposed a fast coordinate descent method with the best variable selection to reduce the objective function. The algorithm iteratively selects variables to update the approximate solution until the accelerated stopping condition $max_{ij} J_{ij}^F < \epsilon p_{\text{init}}$ satisfied, where J_{ij}^F is the reduction of the objective function based on the variable F_{ij}, and p_{init} is the maximum initial reduction over the matrix F. Although the greedy update method does not have guaranteed convergence, it is one of the fastest methods in many reports.

Subsequently, Gillis and Glineur [23] proposed a number of accelerated algorithms using fast approximation by fixing all variables but excepting a single column of factor matrices. This framework improved significantly the effectiveness of multiplicative updates [29], hierarchical alternating least squares (HALS) algorithms [30] and projected gradients [17]. These algorithms achieve the accelerated condition in each iteration such as that $\|F^{(k,l+1)} - F^{(k,l)}\|_2^2 \leq \epsilon \|F^{(k,1)} - F^{(k,0)}\|_2^2$ is the stopping condition when optimizing the objective function on F if fixing W. Although these greedy algorithms does not have guaranteed convergence, their results are highly competitive with the inexact and exact methods.

More recently, Guan et al. [24] employed Nesterov's optimal methods to optimize NNLS with fast convergence of $\mathcal{O}(1/k^2)$ to achieve the accelerated convergence condition $\|\frac{\partial f}{\partial F_{(k,l+1)}}\|_2^2 \leq \epsilon \|\frac{\partial f}{\partial F_{(k,0)}}\|_2^2$. Although Guan et al.'s method converges sub-linearly, it has several drawbacks such as working on the whole factor matrices, and less flexibility for regularized NMF variants. Furthermore, this approach does not consider the issues of parallel and distribution, and they require numerous iterations to satisfy the accelerated condition because the step size is limited by $\frac{1}{L}$, where L is Lipschitz constant.

To deal with these issues of accelerated methods, in the next section, we propose an accelerated parallel and distributed algorithm for NMF and its regularized L_1 L_2 variants with linear convergence in optimizing each factor matrix when fixing the other factor one.

2.3 Related works of nonnegative least squares

In iterative multiplicative update algorithms for NMF, a large number of NNLS or its equivalent problems as NPQ must be solved. The complexity and convergence of these solvers mainly decide the algorithm performance of NMF. Hence, this section briefly reviews the literature of algorithms for NNLS and its equivalent problem NQP (2). In this problem, numerous algorithms are proposed to deal with high dimension [31]. Generally, methods for solving NNLS can divided into two groups: active-set and iterative methods [31]. Active-set methods are traditional to solve accurately [32,33]. However, they require heavy computation in repeatedly computing $(A^T A)^{-1}$ with different set of passive variables. Hence, iterative methods that can handle multiple active constraints in each iteration have more potential for fast NMF algorithms [31,34,35], and these methods can deal with more large-scale problems. Among the fast iterative methods, the coordinate descent method [36] has fast approximation, but it has the zip-zag problem when the solution requires high accuracy. In addition, accelerated methods [37] has a fast convergence $\mathcal{O}(1/k^2)$ [24], which only require the first order derivative. However, one major disadvantage of the methods is that they require a big number of iterations because their step size is limited by $\frac{1}{L}$ that can be very small for large-scale problems; where L is Lipschitz constant. More recently, the accelerated anti-lopsided

algorithm [38] re-scale variables to obtain fast linear convergence $\mathcal{O}([(1-\frac{\mu}{L})(1-\frac{\mu}{rL})^{2r}]^k)$ in the sub-space of passive variables, while the complexity of each iteration is still $\mathcal{O}(r^2)$. Hence, we employ this algorithm to solve NQP problems in NMF.

3 Proposed algorithm

To read easily, our proposed algorithm will be hierarchically presented. First, we introduce an iterative multiplicative update accelerated algorithm, see Algorithm 1. Then, a transformational technique fully decomposes the objective function of NMF into basic computation units as nonnegative quadratic programming (NQP) problems. After that, a modified version of the algorithm is proposed to deal with the issues of parallel and distributed systems, see Algorithm 2. Subsequently, an accelerated anti-lopsided algorithm is employed to effectively solve NQP problems, see Algorithm 3. Finally, a general extension for L_1 L_2 regularized NMF is discussed.

3.1 Iterative multiplicative update accelerated algorithm

For solving NMF, we employ an iterative multiplicative update accelerated algorithm, like expectation-maximization (*EM*) algorithm, presented in Algorithm 1, which consists of two main steps. The first step, called *E-step*, finds F^+, each column of which F_i^+ is the new representation of a data instance V_i in the new space of latent components W. Meanwhile, the other one, called *M-step*, learns new latent components W^+.

3.2 Full decomposition for NMF

To parallelize and distribute the NMF computation, the objective function of NMF is fully decomposed into non-negative quadratic programming (NQP) problems. In Algorithm 1, the E-step is to find new coordinates of data instances in the space of latent components W by minimizing $J(V\|W^TF) = \|V - W^TF\|_2^2 = \sum_{j=1}^m \|V_j - W^TF_j\|_2^2$. Hence, minimizing $J(V\|W^TF)$ is equivalent to independently minimizing $\|V_j - W^TF_j\|_2^2$ for each instance j since W is fixed. Similarly, the M-step is also equivalent to independently minimizing $\|V_i^T - F^TW_i\|_2^2$ for each feature i, where F is fixed. Hence, the computation is decomposed into computation units as nonnegative least-squares (NNLS) problems [33].

Algorithm 1: Iterative Multiplicative Update Accelerated Algorithm

Input: Data matrix $V = \{V_i\}_{i=1}^m \in \mathbb{R}_+^{n\times m}$ and r.
Output: Latent components $W \in \mathbb{R}^{r\times n}$.

```
begin
    Randomize r nonnegative latent components W ∈ R_+^{r×n} ;
    repeat
        E-step: Fixing W to find F^+ satisfying the accelerated condition;
        M-step: Fixing F to find W^+ satisfying the accelerated condition;
    until Convergence condition is satisfied;
end
```

For large datasets $n, m \gg r$, we equivalently turn these problems into nonnegative quadratic programmings (NQP):

$$\begin{array}{ll} \underset{x}{\text{minimize}} & \frac{1}{2}\|Ax - b\|_2^2 \\ \text{subject to} & x \succeq 0 \in \mathbb{R}^r \\ \text{where} & A \in \mathbb{R}_+^{n \times r}, b \in \mathbb{R}_+^n \end{array} \tag{1}$$

equivalent to

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) = \frac{1}{2}x^T Hx + h^T x \\ \text{subject to} & x \succeq 0 \\ \text{where} & H = A^T A \in \mathbb{R}_+^{r \times r}, h = -A^T b \in \mathbb{R}_+^r \end{array} \tag{2}$$

3.3 Parallel and distributed algorithm using limited internal memory

This section introduces a parallel and distributed algorithm, Algorithm 2, using limited internal memory for learning NMF model W, which is a modified version of Algorithm 1.

For large datasets, the computation can be not performed in a single process, so parallel and distributed algorithm environments are employed to speed up the computation. For parallel and distributed systems, there are three major issues: dependency of computation units, limited internal memory of computing nodes, and accessing external memory. In particular, computation units must be independently conducted as much as possible, since any dependency of these computation units will increase the complexity of implementation and the delay of data transferred over the network that reduces the performance of algorithms. Furthermore, in the parallel distributed systems, the computation units are executed on computing nodes within limited internal memory. Finally, accessing external memory will increase the complexity and reduce the performance.

To solve these issues, the computation is fully parallelized and distributed, which uses limited internal memory in computing nodes. Particularly, Algorithm 2 presents a modified version of iterative multiplicative update algorithms, in which the computation units are fully parallelized and distributed. $Q = WW^T$ is precomputed to reduce the complexity, and finding new coefficients F_j can be independently computed and distributed. In addition, the heaviest computation of $Y = FV^T$ and $H = FF^T$ is divided into computing $F_j V_j^T$ and $F_j F_j^T$ in computing nodes.

The distributed system using MapReduce is presented in Fig. 1, in which data instances and the computation are distributed over the Map nodes. The Reduce nodes sum up the results $F_j F_j^T$ and $F_j V_j^T$ of the Map nodes. Subsequently, in the M-step, the results FF^T and FV^T are employed to compute the new latent components W^+. The M-step can be conducted by a single machine or a distributed system, which depends on the dimension of data instances because the time to distribute this computation over the network is usually considerable.

In comparison, this computing model is much more effective than the previous algorithms [16,39,40] for the following reasons:

- The necessary memory used in computing nodes can be limited in $\mathcal{O}(\text{size}(W, Y, H)) = \mathcal{O}(r(r+n))$. The necessary memory used in the controlled node is $\mathcal{O}(\text{size}(W, Y, H, Q)) = \mathcal{O}(r(r+n))$. In practice, approximate solutions of NQP problems should be cached in external memory to increase accuracy and reduce the number of iterations.

Algorithm 2: Parallel and Distributed Algorithm

Input: Data matrix $V = \{V_j\}_{j=1}^{m} \in \mathbb{R}_+^{n \times m}$ and r.

Output: Latent components $W \in \mathbb{R}_+^{r \times n}$.

begin
  Randomize r nonnegative latent components $W \in \mathbb{R}_+^{r \times n}$;
  repeat
    $Y = \mathbf{0}^{r \times n} \in \mathbb{R}^{r \times n}$ /* $Y = FV^T$ */;
    $H = \mathbf{0}^{r \times r} \in \mathbb{R}^{r \times r}$ /* $H = FF^T$ */;
    $Q = WW^T \in \mathbb{R}^{r \times r}$;
    maxStop $= 0$;
    /*Parallel and distributed*/;
    for $j = 1$ *to* m **do**
      /*call Algorithm 3*/;
      $F_j \approx \underset{x \in \mathbb{R}^r \geq 0}{\arg\min} \left(\frac{1}{2} x^T Q x - (WV_j)^T x\right)$;
      $Y = Y + F_j V_j^T$;
      $H = H + F_j F_j^T$;
    /*Parallel and distributed*/;
    for $i = 1$ *to* n **do**
      /*call Algorithm 3*/;
      $W_i \approx \underset{x \in \mathbb{R}^r \geq 0}{\arg\min} \left(\frac{1}{2} x^T H x - {Y_i}^T x\right)$;
  until *Convergence condition is satisfied*;
end

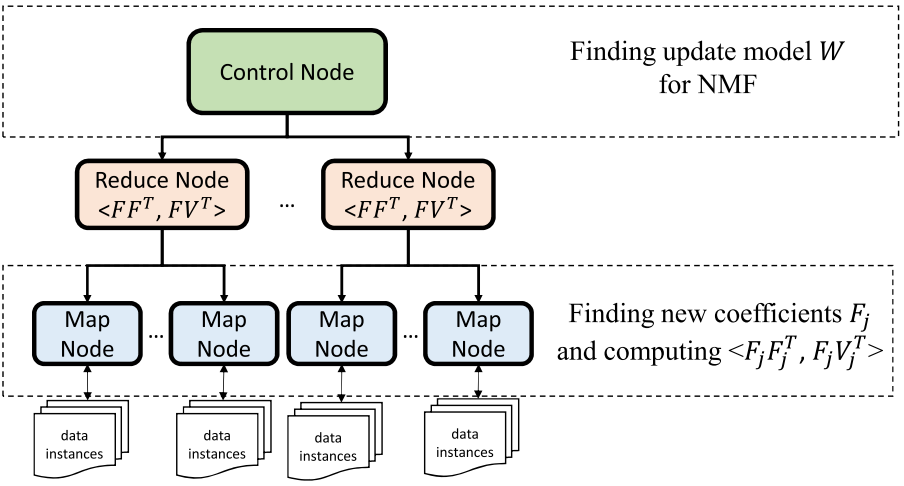


Fig. 1 Distributed system diagram for NMF

– At each distributed iteration, the computation is fully decomposed into basic computation units, which enhances the convergence speed to the optimal solution because the size of optimization problems is significantly reduced. Furthermore, the expensive computation of FV^T and FF^T is fully parallelized and distributed over the computing nodes.

– This computational model is conveniently implemented because its computation is divided into basic computation units as NQP problems that are independently solved, and the optimization is carried out on instance vectors instead of the whole of data matrices.

In the next section, we propose a novel algorithm, Algorithm 3, to approximately solve NQP problems, which is robust and effective because it only uses the first derivative to avoid inverting ill-conditioned matrices.

3.4 Accelerated anti-lopsided algorithm for nonnegative quadratic programming

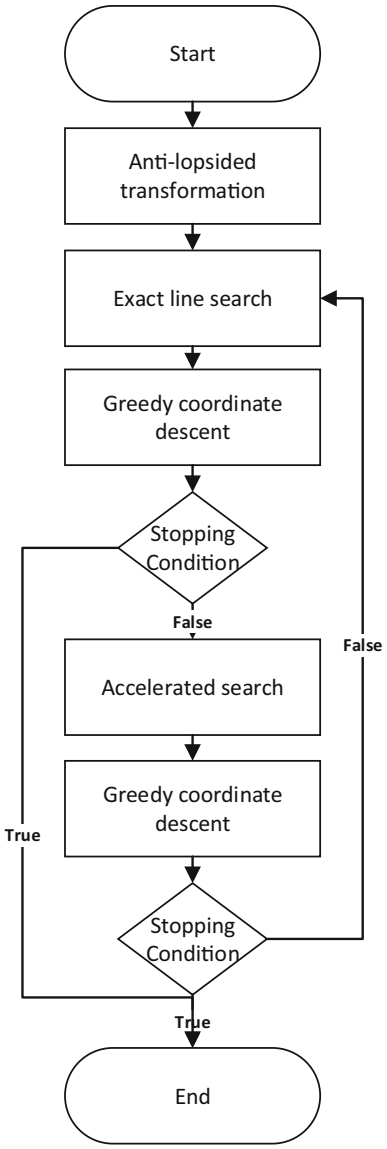
We employ the accelerated anti-lopsided algorithm, the flowchart of which is presented in Fig. 2, having fast linear convergence to reduce the number of iterations, which includes four important parts to attain fast convergence:

– Part 1. Anti-lopsided transformation from Line 3 to Line 3: the variable vector x is transformed into a new space by $x = \varphi(y)$ as an invertible function. In the new space, the new equivalent objective function $g(y) = f(\varphi(y))$ has $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$, or the acceleration of each variable equals 1. As a result, the roles of variables become more balanced because the level curves of the function becomes more spherical since $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$, and $g(y)$ is convex. This part makes the post-processing parts more effective because it can implicitly exploit the second derivative information $\frac{\partial^2 g}{\partial y_i^2} = 1,\ \forall i$ to guarantee that μ and L are always bounded as $\frac{1}{2} \leq \mu \leq L \leq n$.
– Part 2. Exact line search from Line 3 to Line 3: this part optimizes the objective function with a guaranteed and over-bounded convergence rate $\mathcal{O}(1 - \frac{\mu}{L})$, where $\frac{1}{2} \leq \mu \leq L \leq n$ over the space of passive variables. The part reduces the objective function exponentially and precisely, although it suffers from variable scaling problems and negative effects of nonnegative constraints.
– Part 3. Greedy coordinate descent algorithm from Line 3 to Line 3 and repeated in Line 3: this part employs greedy coordinate descent using Gauss-Southwell rule with exact optimization to rapidly reduce the objective function with fast convergence $\mathcal{O}(1 - \frac{\mu}{nL})$ for each update [41,42]. The part reduces negative effects of variable scaling problems and nonnegative constraints, although it has zig-zagging problems. Due to having fast convergence in practice and reducing negative affects of variable scaling problems and nonnegative constraints, this part is repeated one more time after Part 4.
– Part 4. Accelerated search from Line 3 to Line 3: This step performs a momentum search based on the previous changes of variables in Part 2 and Part 3, which has a low complexity of $\mathcal{O}(n.nn(n))$, where $nn(n)$ is the number of negative elements in $(x_{k+1} - \alpha \triangle x)$, see Line 3 in Algorithm 3. This part relies on the global information of two distinct points to escape the local optimal issues of the first derivative raised by the function complexity. This part originates from the idea that if the function is optimized from x_s to x_k by the exact line search and the coordinate descent algorithm, it is highly possible that the function value will be reduced along the vector $(x_k - x_s)$ because the NNLS objective function is convex and has the level curves of eclipse sharp.

Particularly, the anti-lopsided transformation, from Line 3 to Line 3, rescales variables to avoid rescaling problems of the first order methods by replacing $y = x.\sqrt{\text{diag}(H)}$, we have:

$$f(x) = \frac{1}{2} x^T H x + h^T x = \frac{1}{2} y^T Q y + q^T y \tag{3}$$

Fig. 2 Flowchart of algorithm



where $Q = \frac{H}{\sqrt{\text{diag}(H)\text{diag}(H)^T}}$ and $q = \frac{h}{\sqrt{\text{diag}(H)}}$ such that $\frac{\partial^2 f}{\partial^2 y_i} = Q_{ii} = \frac{H_{ii}}{\sqrt{H_{ii}H_{ii}}} = 1$ for $\forall i$. By the way, the acceleration of each variable equals 1, which guarantees that the convex parameter μ and Lipschitz constant L are always bounded as $\frac{1}{2} \leq \mu \leq L \leq r$.

Subsequently, an iteration loop, from Line 3 to Line 3, combines three algorithms including exact line search, greedy coordinate descent, and accelerated search. The accelerated anti-lopsided algorithm guarantees the linear convergence $\mathcal{O}([(1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r}]^k)$ in the sub-space of passive variables to avoid the zip-zag problem of the fast coordinate descent algorithm, while the coordinate block descent algorithm speeds up the convergence to the

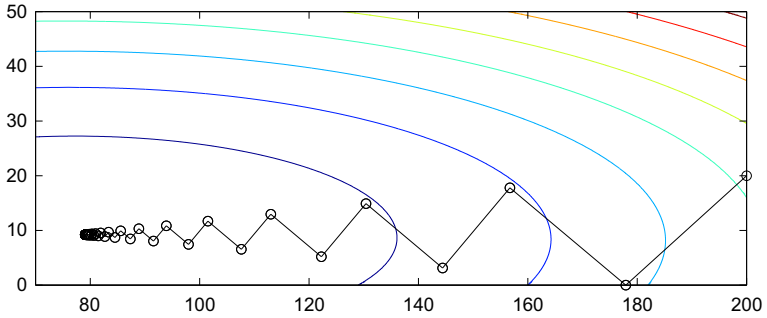


Fig. 3 59 optimizing steps in iterative exact line search method using the first order derivative for the function 4 starting at $x_0 = [200\ 20]^T$

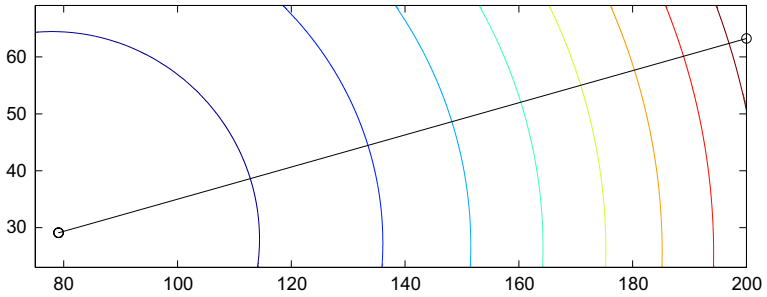


Fig. 4 1 optimizing step in iterative exact line search method using the first order derivative for the function 5 starting at $y_0 = x_0\sqrt{diag(H)}$

final optimal set of passive variables. The passive variables are variables which belong the set $P = \{x_i | x_i > 0 \text{ or } \nabla f_i < 0\}$ that can change through iterations.

In addition, the complexity of each part is still kept in $\mathcal{O}(r^2)$. As a result, the proposed algorithm utilizes the advantages of various algorithms to attain a fast convergence rate, while retaining the same low complexity $\mathcal{O}(r^2)$ of each iteration.

To comprehend the proposed algorithm's effectiveness, consider optimizing function 4:

$$f(x) = \frac{1}{2}x^T \begin{bmatrix} 1 & 0.1 \\ 0.1 & 10 \end{bmatrix} x + [-80 - 100]x \tag{4}$$

The exact search gradient algorithm, from Line 3 to Line 3, starting with $x_0 = [200\ 20]^T$ performs 59 iterations to reach the optimal solution, see Fig. 3. However, the proposed algorithm only needs one iteration to reach the optimal solution, see Fig. 4 because we optimize Function 5 instead of Function 4; where Function 5 is obtained by applying the anti-lopsided transformation from Line 3 to Line 3. The proposed algorithm runs much faster because the level curves of Function 5 become more spherical, and its derivative is more effective to optimize the objective function.

$$f(y) = \frac{1}{2}y^T \begin{bmatrix} 1 & \frac{0.1}{\sqrt{10}} \\ \frac{0.1}{\sqrt{10}} & 1 \end{bmatrix} y + \begin{bmatrix} \frac{-80}{\sqrt{10}} & \frac{-100}{\sqrt{10}} \end{bmatrix} y \tag{5}$$

Algorithm 3 attains approximately accelerated solutions because achieving the optimal solution is controversial for the reasons that its computation is expensive and it can leads to the

Algorithm 3: Accelerated Anti-lopsided Algorithm for NQP

Input: $H \in \mathbb{R}^{r\times r}$ and $h \in \mathbb{R}^r$ and x_0
Output: $x \approx \arg\min_{x\geq 0} \frac{1}{2}x^T Hx + h^T x$

begin
  /*Having a variable maxStop = 0 for each thread of computation */;
  /*Re-scaling variables*/;
  $Q = \frac{H}{\sqrt{\text{diag}(H)\text{diag}(H)^T}}; q = \frac{h}{\sqrt{\text{diag}(H)}};$
  /*Solving NQP: minimizing $f(x) = \frac{1}{2}x^T Qx + q^T x$*/;
  $x = x_0 . * \sqrt{\text{diag}(H)};$
  $\nabla f = Qx + q;$
  repeat
    $x_s = x_{k-1}$ and $\nabla f_s = \nabla f$;
    /*Exact line search over passive variables*/;
    $\nabla \bar{f} = \nabla f$; and $\nabla \bar{f}[x = 0 \text{ and } \nabla f > 0] = 0;$
    $\alpha = \arg\min_{\alpha} f(x_k - \alpha\nabla \bar{f}) = \frac{\|\nabla \bar{f}\|_2^2}{\nabla \bar{f}^T Q \nabla \bar{f}};$
    $x_k = x_{k-1} - \alpha\nabla \bar{f}; \nabla f_k = \nabla f_k - \alpha Q\nabla \bar{f} - Q[x_k]_-; x_k = [x_k]_+;$
    /*Greedy coordinate descent algorithm*/;
    for *t=1 to n* **do**
      $p = \arg\max_{i\in P(x)} |\nabla_i f(x_k)|;$
      $\triangle x_p = max(0, [x_k]_p - \frac{\nabla_p f}{Q_{pp}}) - [x_k]_p;$
      $\nabla f = \nabla f + Q_p \triangle x_p; [x_k]_p = [x_k]_p + \triangle x_p;$
    if $(\|\tilde{f}_k\|_2^2 \leq \epsilon\|\tilde{f}_0\|_2^2)$ *or* $(\|\tilde{f}_k\|_2^2 \leq maxStop)$ **then**
      break;
    /*Accelerated search carries a "momentum" based on the changes of variables in exact line search and greedy coordinate descent part*/;
    $\triangle x = x_s - x_k$ /*x_s and ∇f_s are assigned in Line 3*/;
    $\alpha = \arg\min_{\alpha} f(x_k - \alpha\triangle x) = \frac{\nabla f^T \triangle x}{\triangle x^T Q \triangle x} = \frac{\nabla f^T \triangle x}{\triangle x^T (\nabla f_s - \nabla f)};$
    $x_k = x_k - \alpha\triangle x;$
    $\nabla f = \nabla f - \alpha Q\triangle x - Q[x_k]_-; x_k = [x_k]_+;$
    Repeat steps in the part of greedy coordinate descent algorithm;
  until $(\|\tilde{f}_k\|_2^2 \leq \epsilon\|\tilde{f}_0\|_2^2)$ *or* $(\|\tilde{f}_k\|_2^2 \leq maxStop)$;
  maxStop = max(maxStop, $\|\tilde{f}_k\|_2^2$);
  return $\frac{x_k}{\sqrt{\text{diag}(H)}}$
end

zig-zag problem in optimizing a non-convex function. To control and balance the quality of the convergence to the optimal solution, we employ an accelerated condition ($\|\tilde{f}_k\|_2^2 \leq \epsilon\|\tilde{f}_0\|_2^2$) to regulate the quality of the convergence to the optimal solutions of the NQP problems in comparison with initial values and a fast-break condition ($\|\tilde{f}_k\|_2^2 \leq$ maxStop) to balance the quality of the convergence among variables in each thread of the computation. As a result, the objective function converges faster through iterations, and the average of the iteration numbers are significantly reduced.

3.5 Extensions for L_1 L_2 regularized NMF

To enhance the quality of NMF, we propose a general solution for L_1 L_2 regularized NMF variants, which can achieve sparse solutions. Usually, only the coefficient matrix F is penalized to control its sparsity. Meanwhile, concerning L_2 regularized NMF, the penalty terms of F and W control the smoothness of solutions in NMF [43]. Fortunately, the objective function of L_1 L_2 regularized NMF can be turned into NQP problems, solutions of which are completely similar to the basic NMF. Particularly, in the most general form, the objective function $J(V\|W^TF)$ is formulated by:

$$\frac{1}{2}\|V - W^TF\|_2^2 + \mu_1\|F\|_1 + \beta_1\|W\|_1 + \mu_2\|F\|_2^2 + \beta_2\|W\|_2^2 \tag{6}$$

where $\|.\|_1$ is the L_1-norm, $\|.\|_2$ is the L_2-norm, and $\mu_1, \mu_2, \beta_1, \beta_2$ are regularized parameters that tradeoff the sparsity and the smoothness. Obviously, both the E-step and the M-step need to solve the similar NNLS problems when fixing one of the two matrices. For example, in a E-step, we can minimize the objective function by independently solving NQP problems when fixing W:

$$\begin{aligned}
&\frac{1}{2}\|V - W^TF\|_2^2 + \mu_1\|F\|_1 + \mu_2\|F\|_2^2 + \mathrm{C} \\
&= \sum_{j=1}^{m}\left(\frac{1}{2}\|V_j - W^TF_j\|_2^2 + \mu_1\left(1^K\right)^T F_j + \mu_2 F_j^T I F_j\right) + \mathrm{C} \\
&= \sum_{j=1}^{m}\left(\frac{1}{2}F_j^T Q F_j + q^T F_j\right) + \mathrm{C}
\end{aligned} \tag{7}$$

where $Q = WW^T + 2\mu_1 I$, $q^T = -WV_j + \mu_2 1^K$ and C is a constant.

This transformation from minimizing the objective functions into solving NQP problems is comprehensive to understand and simplify the variants of L_1 L_2 regularized NMF. As a result, we can conveniently implement NMF and its L_1 L_2 regularized variants in parallel distributed systems as in Sect. 3.3.

In comparison with the previous algorithms that performs on the whole of matrices, this approach decomposing the objective function is easier to parallelize and distribute the computation. Additionally, it is faster to reach the solutions because it only performs on a smaller set of variables.

4 Theoretical analysis

This section investigates the convergence of Algorithm 3 and the complexity of Algorithm 2 using Algorithm 3.

4.1 Convergence

This section only discusses the convergence rate of Algorithm 3 for the basic NMF for the two following reasons. Firstly, L_1 regularized coefficients do not affect on the convergence. Secondly, L_2 regularized coefficients are often small, and they change Lipschitz constants μ and L by adding a small positive value, where μ and L are the convex parameter and the Lipschitz constant of strongly convex function $f(x)$ satisfying $\mu I \preceq \frac{\partial^2 f}{\partial^2 x} \preceq LI$, and I is

the identity matrix. Hence, L_2 regularized coefficients slightly change the convergence rate because it depends on $\frac{\mu}{L}$.

Considering the complexity of Algorithm 3, we have:

Lemma 1 *The exact line search in Algorithm 3 linearly converges at an over-bounded rate* $(1 - \frac{\mu}{L})$ *in the sub-space of passive variables, where* $\frac{1}{2} \leq \mu \leq L \leq r$, r *is the number of latent components.*

Proof Based on Theorem 4.5 and Section 4.1.4 in Lecture 4 of [44], we have, we have:

Remark 1 After $(k+1)$ iterations, $f(x^{k+1}) - f^* \leq (1 - \frac{\mu}{L})^k (f(x^0) - f^*)$, where $\mu I \preceq \nabla^2 f \preceq LI$, f^* is the minimum value of $f(x)$, and $f(x)$ is a strongly convex function of the passive variables.

We have $\nabla^2 f = Q$, and $\frac{1}{2} x^T I x \leq \sum_{i=1}^{r} \sum_{j=1}^{r} Q_{ij} x_i x_j = x^T Q x, \forall x$ since $Q_{ij} = \cos(W_i^T, W_j^T)$, and $Q_{ii} = 1. \Rightarrow \frac{1}{2} I \preceq Q$.

Moreover, based on Cauchy–Schwarz inequality, we have:

$$\left(\sum_{i=1}^{r} \sum_{j=1}^{r} Q_{ij} x_i x_j \right)^2 \leq \left(\sum_{i=1}^{r} \sum_{j=1}^{r} Q_{ij}^2 \right) \left(\sum_{i=1}^{r} \sum_{j=1}^{r} (x_i x_j)^2 \right)$$

$$\Rightarrow \sum_{i=1}^{r} \sum_{j=1}^{r} Q_{ij} x_i x_j \leq \sqrt{\|Q\|_2^2 \left(\sum_{i=1}^{r} {x_i}^2 \right)^2}$$

$$\Leftrightarrow x^T Q x \leq \|Q\|_2 x^T I x \quad (\forall x) \quad \Leftrightarrow Q \preceq \|Q\|_2 I$$

Finally, $\sqrt{r} = \sqrt{\sum_{i=1}^{r} Q_{ii}^2} \leq \|Q\|_2 = \sqrt{\sum_{i=1}^{r} \sum_{j=1}^{r} Q_{ij}^2} \leq \sqrt{r^2} = r$ since $-1 \leq Q_{ij} = \cos(W_i^T, W_j^T) \leq 1$. Therefore, we have:

Remark 2 $\frac{1}{2} \leq \mu \leq L \leq r$.

From Remark 1 and Remark 2, we have Lemma 1. □

Actually, the exact line search step, from Line 3 to Line 3 in Algorithm 3, guarantees linear convergence of $(1 - \frac{\mu}{L})$ in the sub-space of passive variables. However, the set of passive variables changes through iterations. Hence, we employ $2r$ times of the greedy coordinate descent update, which also has a fast convergence rate $(1 - \frac{\mu}{2L})^{2r}$ [41,42]. Hence, the greedy coordinate descent algorithm rapidly restricts the domain of solution to converge to the final optimal sub-space of passive variables of the solution. Hence, the proposed algorithm linearly converges at a rate $((1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r})$, where μ and L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$. Therefore, the convergence rate is over-bounded $(1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r} \leq (1 - \frac{1}{2n})(1 - \frac{1}{2rn})^{2r}$. Hence, we have:

Theorem 1 *Algorithm 3 linearly converges at an over-bounded linear rate* $(1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r}$ *in the sub-space of passive variables, where* $\frac{1}{2} \leq \mu \leq L \leq r$, $(1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r} \leq (1 - \frac{1}{2n})(1 - \frac{1}{2rn})^{2r}$, r *is the number of latent components.*

4.2 Complexity

This section analyzes the complexity of Algorithm 2 using Algorithm 3 to solve NQP problems. If we assume that the complexity for each iteration contains $\mathcal{O}(nr^2)$ in computing

Table 2 Complexity of an iteration in NMF solvers

Solver	Complexity ($\mathcal{O}$)
MUR [29]	$mnr + (m+n)r^2$
PrN [27]	$mnr + (m+n)r^2 + r^3$
PrG [17]	$(m+n)r^2 + rmn + \bar{k}\bar{t}(m+n)r^2$
PQN [19]	$\bar{k}(mnr + m^3r^3 + n^3r^3)$
BlP [21]	$(m+n)r^2 + mnr + \bar{k}(m+n)r^2$
AcS [20]	$(m+n)r^2 + rmn + \bar{k}(m+n)r^2$
FCD [22]	$(m+n)r^2 + rS(mn) + \bar{k}(m+n)r^2$
AcH [23]	$(m+n)r^2 + rS(mn) + \bar{k}(m+n)r^2$
Nev [24]	$(m+n)r^2 + mnr + \bar{k}(m+n)r^2$
Alo	$(m+n)r^2 + rS(mn) + \bar{k}(m+n)r^2$

where $m \times n$ is the matrix size, r is the number of latent components, $\bar{k}$ is the average number of iterations, $\bar{t}$ is the average of internal iterations, and $S(mn)$ is the number of non-zero elements of data matrix V. To easily compare among the algorithms, we consider r update times for Algorithm FCD as one iteration because the complexity of one update is $\mathcal{O}(r)$, while the complexity of one iteration in other accelerated algorithms is $\mathcal{O}(r^2)$

$Q = WW^T$, $\mathcal{O}(mnr)$ in computing $Y = VF^T$, $\mathcal{O}(mr^2)$ in computing $H = FF^T$, $\mathcal{O}(\bar{k}mr^2)$ in computing F and $\mathcal{O}(\bar{k}nr^2)$ in computing W, where $\bar{k}$ is the number of iterations, then we have the following Lemma 3:

Theorem 2 *The complexity of each iteration in Algorithm 2 using Algorithm 3 to solve NQP problems is* $\mathcal{O}((m+n)r^2 + mnr + \bar{k}(m+n)r^2)$. *In addition, it is* $\mathcal{O}((m+n)r^2 + rS(mn) + \bar{k}(m+n)r^2)$ *for sparse data, where* $S(mn)$ *is the number of non-zero elements in data matrix* V.

Theorem 2 is significant for big data which are usually big and sparse. For these data, mn is actually large, but $S(mn)$ is small; so $mn \gg (m+n)r^2 + rS(mn) + \bar{k}(m+n)r^2$. Hence, in experimental evaluation Sect. 5, we indicate that our algorithm can run on large high-dimension sparse datasets such as Nytimes for an acceptable time. In that dataset, $mnr \gg rS(mn) \gg (m+n)r^2$, so the running time $T(m,n,r) \approx rS(mn)$ since $m, n \gg r$.

Moreover, Table 2 shows a comparison of the complexity in an iteration of our proposed algorithms (Alo) with other state-of-the-art algorithms' in the literature: Multiplicative Update Rule (MUR) [29], Projected Nonnegative Least Squares (PrN) [27], Projected Gradient (PrG) [17], Projected Quasi-Newton (PQN) [19], Active Set (AcS) [20], Block Principal Pivoting (BlP) [21], Accelerated Hierarchical Alternating Least Squares (AcH) , Fast Coordinate Descent Methods with Variable Selection (FCD) [22], and Nesterov's Optimal Gradient Method (Nev) [24]. Based on the table, the complexity of our proposed algorithm is highly comparable with the other algorithms. In addition, the speed of algorithms mainly depends on the number of iterations. Hence, the result of the experimental evaluation also shows that the iteration number of our algorithm is highly competitive with other algorithms'. Moreover, our proposed algorithm has the following properties, which is necessary for analyzing big data, that other algorithms have yet considered:

Table 3 Dataset information

Datasets	m	n	r	MaxIter
Faces	6977	361	60	300
Digits	6.10^4	784	80	300
Tiny images	5.10^4	3072	100	300
Nytimes	3.10^5	102,660	100,...,200	300

– Exploit the sparseness of datasets,
– Fully decompose the objective function into optimization problems with the smallest size,
– Runnable for big datasets in limited internal memory systems,
– Convenient to implement in fully paralleled and distributed systems.

5 Experimental evaluation

To investigate the effectiveness of the proposed algorithm **Alo**, we compare it to seven carefully selected state-of-the-art NMF solvers belonging to different approaches:

– **MUR** Multiplicative Update Rule [5],
– **PrG** Projected Gradient Methods [17],
– **BlP** Block Principal Pivoting method [21],
– **AcS** Fast Active-set-like method [20],
– **FCD** Fast Coordinate Descent methods with variable selection [22],
– **AcH** Accelerated Hierarchical Alternating Least Squares [23],
– **Nev** Nesterov's optimal gradient method [24].

Test cases We design two test cases using four datasets shown in Table 3. In the first test case, three typical datasets with different sizes are used: Faces[1], Digits[2] and Tiny Images. [3] For these datasets, the algorithms are compared in terms of convergence, optimality, and average of the iteration number to investigate their performance and effectiveness. Additionally, the average of the iteration number $\bar{k}$ for approximate solutions of the sub-problems as NNLS or NQP is used to compare the complexity of algorithms. In the second test case, a large dataset containing tf-idf values computed from the text dataset Nytimes[4] is used to verify the performance and the feasibility of our parallel algorithms on sparse large datasets.

Environment settings To be fair in comparison, for the first test, the programs of compared algorithms are written in the same language Matlab 2013b, run by the same computer Mac Pro 8-Core Intel Xeon E5 3 GHz RAM 32 GB, and initialized by the same factor matrices W_0 and F_0. The maximum number of threads is set to 10 while keeping 2 threads for other tasks in the operation system. For the second test, the proposed algorithm is written in Java programming language to utilize the data sparseness.

[1] http://cbcl.mit.edu/cbcl/software-datasets/FaceData.html.

[2] http://yann.lecun.com/exdb/mnist/.

[3] http://horatio.cs.nyu.edu/mit/tiny/data/index.html.

[4] https://archive.ics.uci.edu/ml/machine-learning-databases/bag-of-words/.

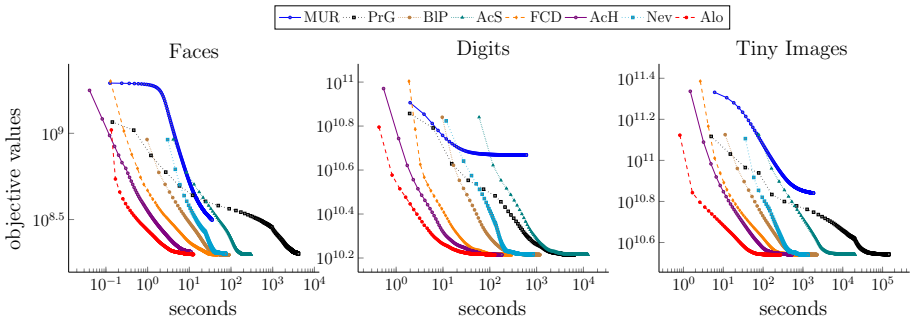


Fig. 5 Objective function values $\|V - W^T F\|_2^2/2$ versus CPU seconds for datasets: faces, digits, and tiny images

Source code The source codes of, **MUR**, **PrG**, **BlP**, **AcS**, **FCD**, **AcH** and **Nev** are downloaded from.[5,6,7,8,9] For convenient comparison in the future, we publish all the source codes and datasets in.[10]

5.1 Convergence

We investigate the convergence of algorithms by information loss $\frac{1}{2}\|V - W^T F\|_2^2$ in terms of time and the iteration number. In terms of time, see Fig. 5, the proposed algorithm Alo is remarkably faster than the other algorithms for the three different-size datasets: Faces, Digits and Tiny Images. Specially, for the largest dataset Tiny Images, the distinction between the proposed algorithm and the runner-up algorithm AcH is easily recognized. Furthermore, in terms of the iteration number, see Fig. 6, the proposed algorithm converges to the stationary point of solutions faster than the others. This observation is clear for large datasets as Digits and Tiny Images. The results are significant in learning NMF models for big data because the proposed algorithm not only converges faster but also uses a less number of iterations, and the time of reading and optimization through a big dataset is actually considerable.

5.2 Optimality

After more than a decade of rapid development, numerous algorithms have been proposed for solving NMF as a fundamental problem in dimension reduction and learning representation. Currently, the difference of the final loss information $\|V - W^T F\|_2^2$ among the state-of-the-art methods is inconsiderable in comparison to the square of information $\|V\|_2^2$. However, this small difference can represent the effectiveness of the optimization methods because NMF algorithms often slowly converge when the approximate solution is close to the optimal local solution. Hence, Table 4 shows the final values of the objective function $\frac{1}{2}\|V - W^T F\|_2^2$ to investigate the optimality and the effectiveness of the compared algorithms. Noticeably, Algorithm AcH fast converges over time and has a low average of the iteration number,

[5] http://www.cs.toronto.edu/~dross/code/nnmf.m.

[6] https://github.com/kimjingu/nonnegfac-matlab.

[7] http://www.csie.ntu.edu.tw/~cjlin/nmf/.

[8] http://dl.dropboxusercontent.com/u/1609292/Acc_MU_HALS_PG.zip.

[9] https://sites.google.com/site/nmfsolvers/.

[10] https://github.com/khuongnd/NMF_APD.

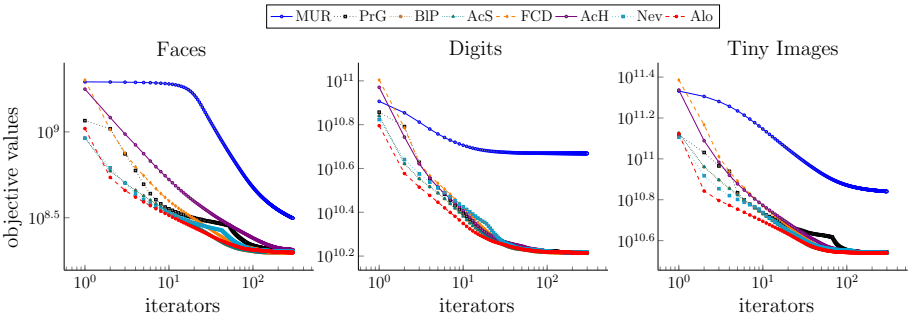


Fig. 6 Objective function values $\|V - W^T F\|_2^2/2$ in terms of the iteration number for datasets: faces, digits, and tiny images

Table 4 Optimal values of NMF solvers

Dataset	MUR	PrG	BlP	AcS	FCD	AcH	Nev	Alo
Faces (10^8)	3.142	2.003	**1.975**	**1.975**	1.983	2.058	2.003	1.985
Digits (10^{10})	4.659	1.639	1.641	1.641	1.644	1.640	1.646	**1.639**
Tiny images (10^{10})	6.925	3.483	3.472	3.472	3.474	3.476	3.473	**3.467**

The most optimal values are shown in bold

Table 5 Average of iteration number $\bar{k}$

Dataset	MUR	PrG	BlP	AcS	FCD	AcH	Nev	Alo
Faces	1.00	321.12	1116.96	102.09	1.54	1.11	29.21	**1.01**
Digits	1.00	36.70	12503.75	305.94	1.00	1.05	23.36	**1.00**
Tiny images	1.00	767.45	12869.12	1086.51	1.38	2.52	29.32	**1.02**

The lowest results except for MUR's ones are shown in bold

but it has the optimal values much higher than the proposed algorithm because it is not guaranteed, and it uses a time-break technique to interrupt the optimization algorithm. The proposed algorithm achieves the best optimality for two largest of three datasets. This result additionally indicates the robustness of the proposed algorithm, which is highly competitive with the state-of-the-art methods.

5.3 Average of iteration number

We investigate the complexity of the NMF solvers by the average of the iteration numbers $\bar{k} = \frac{\text{number of internal iteration}}{\text{MaxIter} \times (m+n)}$ for approximate solutions of sub-problems as NNLS or NQP because the complexity of algorithms mainly depends on this number, see Table 2. Except for the original algorithm MUR with one update having the worst result, the proposed algorithm Alo employs at least the average of the iteration numbers, see Table 5, especially for large datasets. In addition, the proposed algorithm does not employ any tricks to timely interrupt before one of the stopping conditions is satisfied, while the highly competitive algorithm AcH uses. Therefore, this result clearly represents the fast convergence of Algorithm 3 since it is verified by a large number of NQP problems.

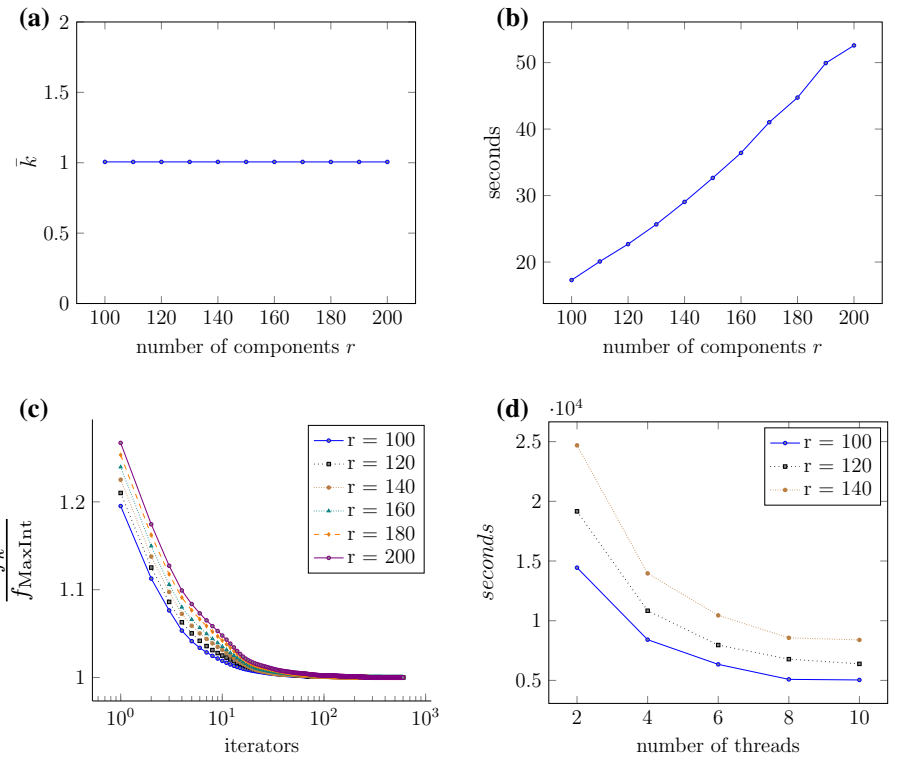


Fig. 7 **a** Average of the iteration number $\bar{k}$, **b** average of iteration time, **c** convergence of $\frac{f_k}{f_{\text{MaxInt}}}$ in learning NMF model within the different numbers of latent components, and **d** running time with different numbers of threads for the dataset Nytimes

5.4 Running on large datasets

We verify the feasibility of the proposed algorithm in learning NMF model for large datasets. Particularly, the proposed algorithm is implemented by Java programming language to exploit the data sparseness. Additionally, it runs on the large sparse text dataset Nytimes with different numbers of latent components and using different number of threads, see Table 3. Interestingly, the proposed algorithm can run with hundreds of latent components by a single computer in an acceptable time.

Figure 7 shows the performance of our algorithm running on the large sparse dataset Nytimes. Remarkably, the proposed algorithm only uses about one iteration on average to satisfy the accelerated condition of approximate solutions, see Fig. 7a. Furthermore, the average of the iteration time in learning NMF model linearly increases through the different numbers of latent components, see Fig. 7b. This result totally fits the complexity analysis when $rnm \gg rS(mn) \gg (m+n)r^2 + \bar{k}(m+n)r^2$, so the complexity $T(m,n,r) \approx rS(mn)$ since $m, n \gg r$. Additionally, the objective function converges to the stationary point at about the 100th iteration within the different numbers of latent components r, which is the same with the previous datasets, see Fig. 7c. Finally, the running time significantly and steadily decreases when increasing the number of used threads, see Fig. 7d. This result represents the effectiveness and stability of the proposed algorithm for parallel computing, which totally

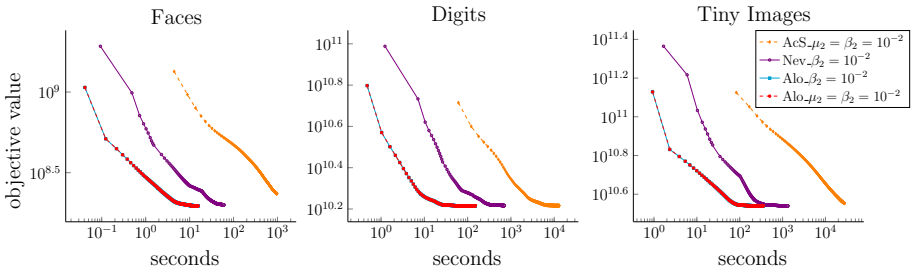


Fig. 8 Convergence of regularized NMF extensions for algorithms AcS, Nev and Alo within two regularized cases: $\mu_2 = 10^{-2}$ and $\mu_2 = \beta_2 = 10^{-2}$

fits with the independence analysis of the full decomposition of optimizing the objective function.

5.5 Regularized NMF extensions

We investigate the convergence of algorithms for regularized NMF extensions on three datasets: Faces, Digits, and Tiny Images. Due to the lack of available codes and the L_1 L_2 generalization of the other algorithms, only three algorithms AcS, Nev and Alo are compared within two regularized cases: $\mu_2 = 10^{-2}$ and $\mu_2 = \beta_2 = 10^{-2}$, see Fig. 8. In comparison with other algorithms for regularized NMF extensions, the proposed algorithm Alo converges much faster than the other algorithms AcS and Nev.

6 Conclusion and discussion

We propose a general flexible algorithm in a unified framework for NMF and its L_1 L_2 regularized variants based on the full decomposition and a fast accelerated anti-lopsided algorithm for NMF. The proposed algorithm linearly converges at an over-bounded rate $(1 - \frac{\mu}{L})(1 - \frac{\mu}{rL})^{2r}$ in optimizing each matrix factor in the sub-space of passive variables when fixing the other matrix, where μ and L are bounded as $\frac{1}{2} \leq \mu \leq L \leq r$ and r is the number of latent components. The proposed algorithm is advanced compared to fast block coordinate descent methods and accelerated methods. In theory and practice, the proposed algorithm resolves some current major issues of NMF: fast guaranteed learning algorithm, data sparseness exploitability, and parallel distributed feasibility using limited internal memory. Furthermore, the proposed algorithm flexibly adapts with the most general variant of L_1 L_2 NMF regularizations.

In the experimental evaluation, our algorithm outperforms seven of the most state-of-the-art algorithms in large datasets about three significant aspects of convergence, average of the iteration number and optimality. Additionally, it can be fully parallelized and distributed because the computation using limited internal memory can be fully decomposed into the basic computation units as NQP problems. Concerning the feasibility in real applications, the proposed algorithm exploits the data sparseness to learn the huge sparse dataset Nytimes in an acceptable time by a single machine. Finally, the convergence of the proposed algorithm for L_1 L_2 regularized NMF variants is much faster than that of the existing algorithms.

Concerning the optimization techniques for alternating least squares methods, we propose a fast algorithm, Algorithm 3 for NQP problems, which not only linearly converges at an

over-bounded rate in theory but also is verified in practice about the three significant aspects by a large number of NQP problems conducted inside the NMF framework. Hence, we strongly believe that the algorithm can be effectively employed for alternating least square methods as the key problem in factorization methods. Hence, in further researches, we will generalize the proposed algorithm for nonnegative tensor factorization.

Acknowledgments This work was supported by Asian Office of Aerospace R&D under agreement number FA2386-15-1-4006, 911 Scholarship from Vietnam Ministry of Education and Training, and National University at Ho Chi Minh City (VNU-HCMC) under the grant number B2015-42-02.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Zhang, Z.Y.: Nonnegative matrix factorization: models, algorithms and applications. In: Holmes, D.E., Jain, L.C. (eds.) Data mining: foundations and intelligent paradigms: volume 2: statistical, bayesian, time series and other theoretical aspects, pp 99–134. Springer, Berlin, Heidelberg (2012). doi:10.1007/978-3-642-23241-1_6
2. Helén, M., Virtanen, T.: Separation of drums from polyphonic music using non-negative matrix factorization and support vector machine. In: Proceedings of the EUSIPCO, vol. 2005 (2005)
3. Donoho, D., Stodden, V.: When does non-negative matrix factorization give a correct decomposition into parts? In: Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference. MIT Press (2004)
4. Gillis, N.: The why and how of nonnegative matrix factorization. Regul. Optim. Kernels Support Vector Mach. **12**, 257 (2014)
5. Lee, D., Seung, H., et al.: Learning the parts of objects by non-negative matrix factorization. Nature **401**(6755), 788–791 (1999)
6. Paatero, P., Tapper, U.: Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values. Environmetrics **5**(2), 111–126 (1994)
7. Wang, Y.X., Zhang, Y.J.: Nonnegative matrix factorization: a comprehensive review. IEEE Trans. Knowl. Data Eng. **25**(6), 1336–1353 (2013). doi:10.1109/TKDE.2012.51
8. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Leen, T.K., Dietterich, T.G., Tresp, V. (eds.) Advances in neural information processing systems, pp. 556–562. MIT Press (2001). http://papers.nips.cc/paper/1861-algorithmsfor-non-negative-matrix-factorization.pdf
9. Zhang, Z.Y.: Divergence functions of non negative matrix factorization: a comparison study. Commun. Stat. Simul. Comput. **40**(10), 1594–1612 (2011)
10. Hoyer, P.O.: Non-negative matrix factorization with sparseness constraints. J. Mach. Learn. Res. **5**, 1457–1469 (2004)
11. Pascual-Montano, A., Carazo, J.M., Kochi, K., Lehmann, D., Pascual-Marqui, R.D.: Nonsmooth nonnegative matrix factorization (nsnmf). IEEE Trans. Pattern Anal. Mach. Intell. **28**(3), 403–415 (2006)
12. Choi, S.: Algorithms for orthogonal nonnegative matrix factorization. In: Neural Networks, 2008. IJCNN 2008. IEEE World Congress on Computational Intelligence, IEEE International Joint Conference on, pp. 1828–1832. IEEE (2008)
13. Li, H., Adal, T., Wang, W., Emge, D., Cichocki, A.: Non-negative matrix factorization with orthogonality constraints and its application to raman spectroscopy. J. VLSI Signal Process. Syst. Signal Image Video Technol. **48**(1–2), 83–97 (2007)
14. Thurau, C., Kersting, K., Wahabzada, M., Bauckhage, C.: Convex non-negative matrix factorization for massive datasets. Knowl. Inf. Syst. **29**(2), 457–478 (2011)
15. Guan, N., Wei, L., Luo, Z., Tao, D.: Limited-memory fast gradient descent method for graph regularized nonnegative matrix factorization. PloS One **8**(10), e77162 (2013)
16. Liu, C., Yang, H.C., Fan, J., He, L.W., Wang, Y.M.: Distributed nonnegative matrix factorization for web-scale dyadic data analysis on mapreduce. In: Proceedings of the 19th International Conference on World Wide Web, pp. 681–690. ACM (2010)
17. Lin, C.J.: Projected gradient methods for nonnegative matrix factorization. Neural Comput. **19**(10), 2756–2779 (2007)

18. Kim, D., Sra, S., Dhillon, I.S.: Fast newton-type methods for the least squares nonnegative matrix approximation problem. In: SDM, pp. 343–354. SIAM (2007)
19. Zdunek, R., Cichocki, A.: Non-negative matrix factorization with quasi-newton optimization. In: Artificial Intelligence and Soft Computing-ICAISC 2006, pp. 870–879. Springer (2006)
20. Kim, H., Park, H.: Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J. Matrix Anal. Appl. **30**(2), 713–730 (2008)
21. Kim, J., Park, H.: Toward faster nonnegative matrix factorization: a new algorithm and comparisons. In: Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on, pp. 353–362. IEEE (2008)
22. Hsieh, C.J., Dhillon, I.S.: Fast coordinate descent methods with variable selection for non-negative matrix factorization. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1064–1072. ACM (2011)
23. Gillis, N., Glineur, F.: Accelerated multiplicative updates and hierarchical als algorithms for nonnegative matrix factorization. Neural Comput. **24**(4), 1085–1105 (2012)
24. Guan, N., Tao, D., Luo, Z., Yuan, B.: Nenmf: an optimal gradient method for nonnegative matrix factorization. IEEE Trans. Signal Process. **60**(6), 2882–2898 (2012)
25. Kim, J., He, Y., Park, H.: Algorithms for nonnegative matrix and tensor factorizations: a unified view based on block coordinate descent framework. J. Glob. Optim. **58**(2), 285–319 (2014)
26. Lin, C.J.: On the convergence of multiplicative update algorithms for nonnegative matrix factorization. IEEE Trans. Neural Netw. **18**(6), 1589–1596 (2007)
27. Berry, M.W., Browne, M., Langville, A.N., Pauca, V.P., Plemmons, R.J.: Algorithms and applications for approximate nonnegative matrix factorization. Comput. Stat. Data Anal. **52**(1), 155–173 (2007)
28. Bonettini, S.: Inexact block coordinate descent methods with application to non-negative matrix factorization. IMA J. Numer. Anal. **31**(4), 1431–1452 (2011)
29. Lee, D.D., Seung, H.S.: Algorithms for non-negative matrix factorization. In: Advances in Neural Information Processing Systems, pp. 556–562 (2001)
30. Cichocki, A., Zdunek, R., Amari, S.i.: Hierarchical als algorithms for nonnegative matrix and 3d tensor factorization. In: Davies, M.E., James, C.J., Abdallah, S.A., Plumbley, M.D. (eds.) Proceedings of 7th international conference on independent component analysis and signal separation, ICA 2007, pp. 169–176, London, UK, September 9–12, 2007. Springer, Berlin, Heidelberg (2007). doi:10.1007/978-3-540-74494-8_22
31. Chen, D., Plemmons, R.J.: Nonnegativity constraints in numerical analysis. In: Symposium on the Birth of Numerical Analysis, pp. 109–140 (2009)
32. Bro, R., De Jong, S.: A fast non-negativity-constrained least squares algorithm. J. Chemom. **11**(5), 393–401 (1997)
33. Lawson, C.L., Hanson, R.J.: Solving Least Squares Problems, vol. 161. SIAM, Philadelphia (1974)
34. Kim Dongmin, S.S., Dhillon, I.S.: A non-monotonic method for large-scale non-negative least squares. Optim. Methods Softw. **28**(5), 1012–1039 (2013)
35. Kim, D., Sra, S., Dhillon, I.S.: A New Projected Quasi-Newton Approach for the Nonnegative Least Squares Problem. Computer Science Department, University of Texas at Austin, Austin (2006)
36. Franc, V., Hlaváč, V., Navara, M.: Sequential coordinate-wise algorithm for the non-negative least squares problem. In: Gagalowicz, A., Philips, W. (eds.) Proceedings of 11th international conference on computer analysis of images and patterns, CAIP 2005, pp. 407–414, Versailles, France, September 5–8, 2005. Springer, Berlin, Heidelberg (2005). doi:10.1007/11556121_50
37. Nesterov, Y.: A method of solving a convex programming problem with convergence rate $o(1/k^2)$. Sov. Math. Dokl. **27**, 372–376 (1983)
38. Nguyen, D.K., Ho, T.B.: Anti-lopsided algorithm for large-scale nonnegative least square problems. arXiv:1502.01645 (2015)
39. Gemulla, R., Nijkamp, E., Haas, P.J., Sismanis, Y.: Large-scale matrix factorization with distributed stochastic gradient descent. In: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 69–77. ACM (2011)
40. Sun, Z., Li, T., Rishe, N.: Large-scale matrix factorization using mapreduce. In: 2010 IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1242–1248. IEEE (2010)
41. Nesterov, Y.: Efficiency of coordinate descent methods on huge-scale optimization problems. SIAM J. Optim. **22**(2), 341–362 (2012)
42. Schmidt, M., Friedlander, M.: Coordinate descent converges faster with the gauss-southwell rule than random selection. In: NIPS OPT-ML Workshop (2014)
43. Pauca, V.P., Piper, J., Plemmons, R.J.: Nonnegative matrix factorization for spectral data analysis. Linear Algebra Appl. **416**(1), 29–47 (2006)
44. Caramanis, L., Jo, S.J.: EE 381V: Large scale optimization fall 2012. http://sers.ece.utexas.edu/~cmcaram/EE381V_2012F/Lecture_4_Scribe_Notes.final.pdf