NeNMF: An Optimal Gradient Method for Nonnegative Matrix Factorization

Naiyang Guan, Dacheng Tao, *Senior Member, IEEE*, Zhigang Luo, and Bo Yuan

***Abstract*—Nonnegative matrix factorization (NMF) is a powerful matrix decomposition technique that approximates a nonnegative matrix by the product of two low-rank nonnegative matrix factors. It has been widely applied to signal processing, computer vision, and data mining. Traditional NMF solvers include the multiplicative update rule (MUR), the projected gradient method (PG), the projected nonnegative least squares (PNLS), and the active set method (AS). However, they suffer from one or some of the following three problems: slow convergence rate, numerical instability and nonconvergence. In this paper, we present a new efficient NeNMF solver to simultaneously overcome the aforementioned problems. It applies Nesterov's optimal gradient method to alternatively optimize one factor with another fixed. In particular, at each iteration round, the matrix factor is updated by using the PG method performed on a smartly chosen search point, where the step size is determined by the Lipschitz constant. Since NeNMF does not use the time consuming line search and converges optimally at rate $O\left(\frac{1}{k^2}\right)$ in optimizing each matrix factor, it is superior to MUR and PG in terms of efficiency as well as approximation accuracy. Compared to PNLS and AS that suffer from numerical instability problem in the worst case, NeNMF overcomes this deficiency. In addition, NeNMF can be used to solve L_1-norm, L_2-norm and manifold regularized NMF with the optimal convergence rate. Numerical experiments on both synthetic and real-world datasets show the efficiency of NeNMF for NMF and its variants comparing to representative NMF solvers. Extensive experiments on document clustering suggest the effectiveness of NeNMF.**

***Index Terms*—L_1-norm, L_2-norm, manifold regularization, nonnegative matrix factorization (NMF), optimal gradient method.**

I. Introduction

NONNEGATIVE MATRIX FACTORIZATION (NMF) learns an approximate decomposition of a nonnegative data matrix into two low-rank nonnegative matrix factors. Since the nonnegativity constraints on the two matrix factors usually yield sparse representation of the data, NMF has been widely applied to signal processing [6], computer vision [13] and data mining [31], [26]. Recently, several solvers have been proposed for NMF [2], [4], [12], [15], [18], [20], [33]. However, they suffer from some or at least one of the following three problems: slow convergence, numerical instability and theoretical nonconvergence problems. To solve the nonconvex NMF problem, Berry *et al.* [2] proposed an alternating nonnegative least squares (ANLS) framework that treats NMF as two nonnegative least squares (NLS) subproblems and alternatively updates matrix factors based on the solution of the corresponding NLS subproblem until convergence. Based on the matrix updating strategy, representative NMF solvers can be classified into the following two groups:

1) Inexact block coordinate descent (IBCD): IBCD updates matrix factors with approximate solution of the corresponding NLS subproblem. According to the method for approximately solving each NLS subproblem, we categorized the IBCD methods into the following three subgroups:

   Multiplicative update rule (MUR): MUR [18] searches along a rescaled gradient direction with a carefully selected learning rate to guarantee the nonnegativity of the resulted matrix factors. Although MUR is simple and easy to implement, it converges slowly. That is because the used rescaled gradient descent is a first-order method. In addition, MUR does not guarantee the convergence to any local minimum because its solution is unnecessarily a stationary point [21]. To overcome this problem, Lin [21] proposed a modified MUR that converges to a stationary point. However, the modified solver cannot improve the convergence rate.

   Projected NLS method (PNLS): Since MUR updates matrix factors by multiplying each entry with a positive factor in each iteration round, all entries in the obtained matrix factors will not shrink to zero. To overcome this problem, Berry *et al.* [2] presented a projected NLS method (PNLS) which solves each NLS subproblem by directly projecting the least squares solution to the nonnegative quadratic. Note that the PNLS method was first proposed to solve the positive matrix factorization problem [25]. Although PNLS performs well in most applications, it does not guarantee the convergence.

   Cyclic block coordinate gradient projection (CBGP): To overcome the theoretical nonconvergence deficiency of PNLS, Bonettini [4] proposed a CBGP method to solve NMF based on the inexact block coordinate descent

Manuscript received June 07, 2011; revised September 25, 2011 and December 25, 2011; accepted February 22, 2012. Date of publication March 08, 2012; date of current version May 11, 2012. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Konstantinos Slavakis. This work was supported in part by the Australian ARC discovery project (ARC DP-120103730), the National Natural Science Foundation of China (by Grant No. 91024030/G03), and by the Program for Changjiang Scholars and Innovative Research Team in University (No. IRT1012).

N. Guan and Z. Luo are with School of Computer Science, National University of Defense Technology, Changsha, 410073 China (e-mail: ny_guan@nudt.edu.cn; zgluo@nudt.edu.cn).

D. Tao is with Center for Quantum Computation and Intelligent Systems and the Faculty of Engineering and Information Technology, University of Technology, Sydney, NSW 2007, Australia (e-mail: dacheng.tao@uts.edu.au).

B. Yuan is with Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, 200240 China (e-mail: boyuan@sjtu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TSP.2012.2190406

framework. Although CBGP unnecessarily obtains the optimum for both NLS subproblems in each iteration round, it obtains the global convergence because of the Armijo-type line search. We appreciate the global convergence property of CBGP, but it is inefficient, because the Armijo-type line search is time-consuming.

2) Exact block coordinate descent (EBCD): EBCD updates matrix factors with the optimum for both NLS subproblems in each iteration round. According to the methods for optimizing each NLS subproblem, we categorized the EBCD methods into the following two subgroups:
Projected gradient method (PG): Lin [20] deemed each NLS subproblem as a bound constrained optimization problem and applied projected gradient (PG) method to it. In each iteration round, PG uses the Armijo rule to estimate the optimal step size along the projection arc for solving each NLS subproblem. However, PG is inefficient for solving NMF, because the line search procedure is time-consuming. Zdunek and Cichocki [33] applied the projected quasi-Newton method (QN) to each NLS subproblem. QN estimates the step size by using the inverse of Hessian to estimate the optimal step size and thus it converges much faster than PG. However it is time-consuming for calculating the inverse of Hessian and QN will fail when the Hessian is rank deficient. To overcome this problem, Kim *et al.* [17] applied BFGS to the NLS subproblems. BFGS approximates the Hessian based on rank-one updates specified by gradient evaluations. Although BFGS converges, it requires a time-consuming line search and a complex Hessian updating procedure, so it is inefficient. Recently, Han *et al.* [12] presented projected Barzilai-Borwein method (PBB). PBB uses Barzilai and Borwein's strategy to determine the step size and an Armijo-type line search to ensure the global convergence. PBB is also inefficient due to the time-consuming Armijo-type line search.
Active set method (AS): In contrast to the aforementioned gradient-based solver, Kim and Park [16] solved the NLS subproblem by using the active set (AS) method. In particular, AS divides variables into *free set* and *active set*, wherein *free set* contains variables that are not optimal. In each iteration round, AS exchanges one variable between this two sets and solves an unconstrained equation to update the variables in *free set*. Kim and Park [15] further developed the AS method by exchanging multiple variables between *free set* and *active set*, and proposed block principal pivoting (BPP) method that greatly accelerates the convergence of AS. However, both AS and BPP assume each NLS subproblem is strictly convex to make the unconstrained equation solvable. Such assumption may make solvers face the numerical instability problem especially for large scale problems.

In this paper, we propose a new efficient solver NeNMF that simultaneously overcomes the aforementioned problems for NMF. By proving that the objective function of each NLS subproblem is a convex function whose gradient is Lipchitz continuous, we naturally apply Nesterov's optimal gradient method [23] to optimize NLS. In convex optimization, Nesterov's optimal gradient method has been recently applied to several optimization problems appearing in many problems, e.g., compressed sensing [1], trace norm minimization [14] and clustering [34]. In this paper, we utilize this method to optimally minimize one matrix factor with another fixed. In particular, we update two sequences recursively for optimizing each factor, one sequence stores the approximate solutions and another sequence stores the search points. In each iteration round, the approximate solution is obtained by using the PG method on the search point and the step size is determined by the Lipchitz constant, and the search point is constructed by linearly combining the latest two approximate solutions. This scheme greatly accelerates the optimization based on the problem structure. Therefore, this method achieves optimal convergence rate[1] of $O\left(\frac{1}{k^2}\right)$. By alternatively performing the optimal gradient method on the two matrix factors, NeNMF converges much faster than representative NMF solvers. It has neither time-consuming line search procedure nor numerical instability problem comparing against existing solvers.

Preliminary experiments on both synthetic and real-world datasets suggest the efficiency of NeNMF. Document clustering experiments on popular real-world datasets confirm the effectiveness of NeNMF. In addition, we show that NeNMF can be naturally extended to handle several versions of NMF that incorporates L_1-norm, L_2-norm and manifold regularizations. Extensive experiments on synthetic datasets confirm the efficiencies of these extensions.

II. Nonnegative Matrix Factorization Via Optimal Gradient Method

Given n samples $\vec{x}_1, \ldots, \vec{x}_n \in \mathbf{R}_+^m (i = 1, \ldots, n)$ arranged in a nonnegative matrix $X = [\vec{x}_1, \ldots, \vec{x}_n]$, NMF approximates X by two low-rank nonnegative matrices $W \in \mathbf{R}_+^{m \times r}$ and $H \in \mathbf{R}_+^{r \times n}$, i.e.,

$$\min_{W,H} \frac{1}{2}\|X - WH\|_F^2, \quad s.t., \quad W \in \mathbf{R}_+^{m \times r}, \quad H \in \mathbf{R}_+^{r \times n} \tag{1}$$

where $\|\cdot\|_F$ is the matrix Frobenius norm and $r < \min\{m, n\}$.

Since (1) is a nonconvex minimization problem, it is impractical to obtain the optimal solution. Fortunately the block coordinate descent method [3] can obtain a local solution of (1). Given an initial $W^1 \geq 0$, the block coordinate descent method alternatively solves

$$H^{t+1} = \arg\min_{H \geq 0} F(W^t, H) = \frac{1}{2}\|X - W^t H\|_F^2 \tag{2}$$

and

$$W^{t+1} = \arg\min_{W \geq 0} F(W, H^{t+1}) = \frac{1}{2}\left\|X^T - {H^{t+1}}^T W^T\right\|_F^2 \tag{3}$$

until convergence, where t is the iteration counter. Most existing NMF solvers are special implementations under this scheme. Different solvers use different optimization methods to minimize (2) and (3). Since problems (2) and (3) are symmetric, we only need to show how to efficiently solve (2) and (3) can be

[1]In this paper, convergence rate implies the speed of optimizing one matrix factor with another fixed.

solved accordingly. According to *Lemma* 1 and *Lemma* 2, we will show that (2) can be efficiently solved by Nesterov's optimal gradient method (OGM).

Lemma 1: The objective function $F(W^t, H)$ is convex.

Lemma 2: The gradient of the objective function $F(W^t, H)$ is Lipschitz continuous and the Lipschitz constant is $L = \|W^{t^T} W^t\|_2$.

We leave the detailed proofs of *Lemma* 1 and *Lemma* 2 in Appendices A and B, respectively.

A. Optimal Gradient Method

Recent results [22]–[24] in operation research show that the gradient-based methods for smooth optimization can be accelerated and achieve the optimal convergence rate $O\left(\frac{1}{k^2}\right)$. Since $F(W^t, H)$ is convex and the gradient $\nabla_H F(W^t, H)$ is Lipschitz continuous, Nesterov's method can be used to efficiently optimize (2). In particular, we construct two sequences, i.e., $\{H_k\}$ and $\{Y_k\}$, and alternatively update them in each iteration round. At the iteration k, the two sequences are

$$H_k = \arg\min_{H\geq 0} \left\{ \phi(Y_k, H) = F\left(W^t, Y_k\right) + \left\langle \nabla_H F(W^t, Y_k), H - Y_k \right\rangle + \frac{L}{2}\|H - Y_k\|_F^2 \right\}, \quad (4)$$

and

$$Y_{k+1} = H_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(H_k - H_{k-1}) \quad (5)$$

where $\phi(Y_k, H)$ is the proximal function of $F(W^t, H)$ on Y_k, L is the Lipschitz constant given in *Lemma* 2, $\langle \cdot, \cdot \rangle$ denotes the matrix inner product, H_k contains the approximate solution obtained by minimizing the proximal function $\phi(Y_k, H)$ over H, and Y_k stores the search point that is constructed by linearly combining the latest two approximate solutions, i.e., H_{k-1} and H_k. According to [22], the combination coefficient α_{k+1} is carefully updated in each iteration round

$$\alpha_{k+1} = \frac{1 + \sqrt{4\alpha_k^2 + 1}}{2}. \quad (6)$$

By using the Lagrange multiplier method, the Karush-Kuhn-Tucker (K.K.T.) conditions of (4) are

$$\nabla_H \phi(Y_k, H_k) \geq 0, \quad (7)$$

$$H_k \geq 0, \quad (8)$$

$$\nabla_H \phi(Y_k, H_k) \otimes H_k = 0, \quad (9)$$

where $\nabla_H \phi(Y_k, H_k) = \nabla_H F(W^t, Y_k) + L(H_k - Y_k)$ is the gradient of $\phi(Y_k, H)$ with respect to H at H_k, and $\otimes$ is the Hadamard product. By solving (7) to (9), we have

$$H_k = P\left(Y_k - \frac{1}{L}\nabla_H F\left(W^t, Y_k\right)\right) \quad (10)$$

where the operator $P(X)$ projects all the negative entries of X to zero. All variables will not exceed the upper bound by setting it sufficiently large.

By alternatively updating H_k, α_{k+1} and Y_{k+1} with (10), (6) and (5) until convergence, the optimal solution can be obtained. Similar to PG [20], PBB [12] and AS [16], we use the K.K.T. conditions of (2) as the stopping criterion

$$\nabla_H F\left(W^t, H_k\right) \geq 0, \quad (11)$$

$$H_k \geq 0, \quad (12)$$

$$\nabla_H F\left(W^t, H_k\right) \otimes H_k = 0. \quad (13)$$

The K.K.T. conditions (11) to (13) can be re-written as

$$\nabla_H^P F\left(W^t, H_k\right) = 0 \quad (14)$$

where $\nabla_H^P F(W^t, H_k)$ is the projected gradient defined by

$$\nabla_H^P F\left(W^t, H_k\right)_{ij} = \begin{cases} \nabla_H F\left(W^t, H_k\right)_{ij}, & (H_k)_{ij} > 0 \\ \min\left\{0, \nabla_H F(W^t, H_k)_{ij}\right\}, & (H_k)_{ij} = 0. \end{cases}$$

Since the stopping criterion (14) will keep *Algorithm* 1 unnecessarily running, similar to [20], we instead use

$$\begin{aligned} &\left\|\nabla_H^P F\left(W^t, H_k\right)\right\|_F \\ &\leq \epsilon_H \\ &= \max(10^{-3}, \epsilon) \\ &\quad \times \left\| \left[\nabla_H^P F(W^1, H^1), \nabla_W^P F(W^1, H^1)^T\right] \right\|_F \end{aligned}$$

where ϵ_H is the tolerance for (2). If *Algorithm* 1 solves (2) within a predefined minimum step number, i.e., $K_{min}^H = 10$, we adaptively update the tolerance by $\epsilon_H = 0.1\epsilon_H$. The same strategy is adopted to solve (3) by using *Algorithm* 1.

Algorithm 1: Optimal gradient method (OGM)

```
Input: $W^t, H^t$
Output: $H^{t+1}$
1: Initialize $Y_0 = H^t, \alpha_0 = 1, L = \|W^{t^T} W^t\|_2, k = 0$
repeat
    2: Update $H_k$, $\alpha_{k+1}$ and $Y_{k+1}$ with
        2.1 : $H_k = P\left(Y_k - \frac{1}{L}\nabla_H F\left(W^t, Y_k\right)\right)$,    (15)
        2.2 : $\alpha_{k+1} = \frac{1 + \sqrt{4\alpha_k^2 + 1}}{2}$,    (16)
        2.3 : $Y_{k+1} = H_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(H_k - H_{k-1})$.    (17)
    3: $k \leftarrow k + 1$
until Stopping criterion (14) is satisfied
4: $H^{t+1} = H_K$
```

We summarize the optimal gradient method for optimizing (2) in *Algorithm* 1 that accepts input H^t and W^t and outputs H^{t+1}. Both H^t and W^t are obtained from the previous iteration in the block coordinate descent method.

By alternatively updating (15), (16), and (17) until (14) is satisfied, we obtain the optimal solution of (2), namely H_K, wherein K is the iteration number of *Algorithm* 1. The gradient $\nabla_H F(W^t, Y_k) = {W^t}^T W^t Y_k - {W^t}^T X$, wherein $k = \{0, 1, \ldots\}$ is the iteration counter.

In the following *Proposition* 1, we show that the *Algorithm* 1 achieves the optimal convergence rate $O\left(\frac{1}{k^2}\right)$.

Proposition 1: Given sequences $\{H_k\}_{k=0}^{\infty}$ and $\{Y_k\}_{k=0}^{\infty}$ generated by (15) and (17), respectively, we have

$$F\left(W^t, H_k\right) - F\left(W^t, H_*\right) \leq \frac{2L\|H^t - H_*\|_F^2}{(k+2)^2}$$

where H_* is an optimal solution of (2).

We leave the detailed proof in Appendix C.

B. NeNMF

By using *Algorithm* 1 (OGM) to solve (2) and (3), we arrive at NeNMF summarized in *Algorithm* 2. We give the stopping criterion of this algorithm after the algorithm table.

```
Algorithm 2: NeNMF
Input: $V \in \mathbf{R}_+^{m\times n}, 1 \leq r \leq \min\{m, n\}$
Output: $W \in \mathbf{R}_+^{m\times r}, H \in \mathbf{R}_+^{r\times n}$
1: Initialize $W^1 \geq 0, H^1 \geq 0, t = 1$
repeat
    2: Update $H^{t+1}$ and $W^{t+1}$ with
        $2.1: H^{t+1} = OGM(W^t, H^t),$
        $2.2: {W^{t+1}}^t = OGM\left({H^{t+1}}^t, {W^t}^t\right).$
    3: $t \leftarrow t + 1$
until Stopping criterion (21) is satisfied
4: $W = W^t, H = H^t$
```

Similar to *Algorithm* 1, we use the K.K.T. conditions of (1) as a stopping criterion for *Algorithm* 2

$$W^t \geq 0, H^t \geq 0, \tag{18}$$

$$\nabla_H F(W^t, H^t) \geq 0, \nabla_W F(W^t, H^t) \geq 0, \tag{19}$$

$$\nabla_H F(W^t, H^t) \otimes H^t = 0, \nabla_W F(W^t, H^t) \otimes W^t = 0. \tag{20}$$

We reorganize conditions (18) to (20) as

$$\nabla_H^P F(W^t, H^t) = 0, \quad \nabla_W^P F(W^t, H^t) = 0. \tag{21}$$

To check whether (W^t, H^t) is close to a stationary point, according to [20], we transform (21) to

$$\begin{aligned}&\left\|\left[\nabla_H^P F(W^t, H^t), \nabla_W^P F(W^t, H^t)^T\right]\right\|_F \\ &\quad \leq \epsilon \left\|\left[\nabla_H^P F(W^1, H^1), \nabla_W^P F(W^1, H^1)^T\right]\right\|_F, \end{aligned} \tag{22}$$

where ϵ is a tolerance.

TABLE I
TIME COMPLEXITY: NENMF VERSUS THE EXISTING NMF SOLVERS

Solver	Time Complexity
MUR [18]	$O(mnr + mr^2 + nr^2)$
PNLS [2]	$O(mnr + mr^2 + nr^2 + r^3)$
PG [20]	$O(mnr) + K \times O(tmr^2 + tnr^2)$
QN [33]	$O(mnr + m^3r^3 + n^3r^3)$
BFGS [17]	$O(mnr) + K \times O(mnr + tmr^2 + tnr^2)$
AS [16]	$O(mnr + mr^2 + nr^2) + K \times O(mr^2 + nr^2)$
BPP [15]	$O(mnr + mr^2 + nr^2)$ $+K \times O(mr^2 + nr^2 + r^3 + m\log_2 n + n\log_2 m)$
PBB [12]	$O(mnr) + K \times O(tmr^2 + tnr^2)$
CBGP [4]	$O(mnr) + K \times O(tmnr + mr^2 + nr^2)$
NeNMF	$O(mnr + mr^2 + nr^2) + K \times O(mr^2 + nr^2)$

Since the problem (1) is nonconvex, it is impossible to get its global optimum. Existing NMF solvers consider a stationary point as a local optimal solution [4], [12], [15], [20]. In nonlinear optimization, the stationarity of the limit point generated by the block coordinate descent method is guaranteed by the assumption that the optimal solution of each subproblem is unique [3]. However the optimal solution obtained by *Algorithm* 1 is not unique because the problem (2) is not strictly convex. Fortunately, Grippo and Sciandrone [9] have shown that the uniqueness is unnecessary if we have only two blocks, i.e., any limit point of the sequence generated by *Algorithm* 2 is a stationary point of (1). Since the feasible sets of (2) and (3) are bounded given sufficiently large bounds, there exists at least one limit point in the sequence generated by *Algorithm* 2 according to [27]. Therefore, NeNMF converges to a stationary point.

The main time cost of NeNMF is spent on the calculation of the gradient $\nabla_H F(W^t, H_k)$ in *Algorithm* 1. Note that the second term ${W^t}^T X$ in $\nabla_H F(W^t, H_k)$ is a constant and it can be calculated before iterations of *Algorithm* 1. Since $W^t \in \mathbf{R}_+^{m\times r}$ is a constant low-rank matrix, the term ${W^t}^T W^t$ can also be calculated previously in $O(mr^2)$ time. The complexity of one iteration in NeNMF is $O(mr^2 + mnr) + K \times O(nr^2)$. As *Algorithm* 1 converges at rate $O\left(\frac{1}{k^2}\right)$, it stops within a small number of iterations, typically $K < r$. We summarize the time complexity of one iteration round of NeNMF and compare it to those of the representative NMF solvers in Table I, where the variable t in PG [20] and BFGS [17] is the inner iteration number of the line search procedure. Table I shows that the complexity of NeNMF is comparable to the existing NMF solvers. However, it converges much faster at each iteration round and therefore it is faster than others.

C. Related Works

NMF (1) can be solved by alternatively optimizing (2) and (3). However, it is unnecessary to minimize (2) or (3) in each iteration round. For example, the MUR [18] updates the matrix factor H by one step rescaled gradient descent according to

$$H^{t+1} = H^t \otimes \frac{{W^t}^T X}{{W^t}^T W^t H^t}. \tag{23}$$

Guan *et al.* [10], [11] proposed to accelerate MUR by searching each matrix factor along its rescaled gradient direction with the optimal step length.

Since elements in the denominator in (23) can be zero, MUR may fail. To overcome this deficiency, Lin [21] modified MUR by adding a tiny number to each element in the denominator. PNLS [2] overcomes such deficiency by regarding (2) as an NLS and updates H^{t+1} according to

$$H^{t+1} = P\left(W^{t^{\dagger}} X\right) \tag{24}$$

where $W^{t^{\dagger}}$ is the pseudo-inverse of W^t and $P(\cdot)$ is the projection operator defined in (10). PNLS works well in many problems. It is however difficult to analyze the convergence of PNLS, because $P(\cdot)$ may increase the objective function [17].

PG [20] treats (2) as a bound-constrained optimization problem and uses the projected gradient descent to optimize it. In the iteration round t, PG repeats the following update until convergence:

$$H_{k+1} = P\left(H_k - \alpha \nabla_H F\left(W^t, H_k\right)\right) \tag{25}$$

where $\nabla_H F(W^t, H_k)$ is the gradient of $F(W^t, H)$ at H_k and α is the optimal step size estimated by the Armijo rule.

QN [33] regards (2) as a bound-constrained optimization problem. In contrast with PG, QN however uses quasi-Newton method to optimize (2). The update rule of H^{t+1} is

$$H^{t+1} = P\left(H^t - \gamma R_H^t \setminus {Q_H^t}^T \nabla_H F(W^t, H^t)\right) \tag{26}$$

where γ is a relaxation parameter, and $Q_H^t R_H^t = \mathrm{Hessian}_F(H^t) + \lambda I$ is the QR factorization of the regularized Hessian matrix of $F(W^t, H)$ at H^t, i.e., $\mathrm{Hessian}_F(H^t)$, and λ is the tradeoff parameter, $R_H^t \setminus Q_H^t$ is the solution in the least squares sense to the system equation $R_H^t X = Q_H^t$. QN converges fast as it uses the second-order gradient information embedded in Hessian matrix. However QN suffers from the numerical instability problem because the Hessian matrix can be singular. Although the regularization can reduce this problem, it is difficult to choose a proper λ.

Kim *et al.* [17] utilized BFGS to update the Hessian matrix based on the rank-one updates specified by gradient evaluations. In the iteration round t, BFGS iterates the following update until convergence:

$$H_{k+1} = P\left(H_k - diag(\vec{\alpha}_k) D_k \nabla_H F\left(W^t, H_k\right)\right) \tag{27}$$

where $k = 0, 1, \ldots$ is the inner iteration counter, and H is initialized as $H_0 = H^t$. The H is updated by the solution of (27), i.e., $H^{t+1} = H_K$, wherein K is the iteration number of (27). In (27), D_k is an approximation of $Hessian_F^{-1}(H_k)$ that is updated by BFGS, and the step size vector $\vec{\alpha}_k$ is calculated by line search.

Although BFGS solver converges, both Hessian update and line search are complex and time-consuming. Therefore, they proposed an inexact version of BFGS solver by updating H_{k+1} according to

$$H_{k+1} = P\left(H_k - \alpha \left(W^{t^T} W^t\right)^{-1} \nabla_H F\left(W^t, H_k\right)\right) \tag{28}$$

where $W^{t^T} W^t$ is utilized as the approximation of $Hessian_F(H^t)$, and α is a manually prefixed step size. The inner iteration of (28) is repeated for fixed times, e.g., $k = 10$. Then H^{t+1} is updated as $H^{t+1} = H_k$. Though the update (28) converges fast, it is difficult to select a suitable step size for different datasets.

Han *et al.* [12] proposed the projected Barzilai-Borwein method (PBB). PBB approximates the secant equation by using two neighborhood points to determine the step size in (25). The update rule is

$$H_{k+1} = H_k + \lambda_k \nabla_H^P F\left(W^t, H_k\right) \tag{29}$$

where $\nabla_H^P F(W^t, H_k) = P(H_k - \alpha_k \nabla_H F(W^t, H_k)) - H_k$ is the projected gradient with the step size α_k calculated by using Barzilai and Borwein's strategy $\alpha_k = \frac{\langle S,S\rangle}{\langle Y,S\rangle}$, wherein $S = H_k - H_{k-1}$ and $Y = \nabla_H F(W^t, H_k) - \nabla_H F(W^t, H_{k-1})$. Since the objective of (2) is not strictly convex, they introduced an Armijo-type nonmonotone line search for choosing λ_k, and this strategy ensures the global convergence.

Recently, Bonettini [4] proposed a cyclic block coordinate gradient projection (CBGP) method for NMF based on the inexact block coordinate descent. In the iteration round t, CBGP iterates the following update fixed times:

$$H_{k+1} = H_k + \lambda_k \nabla_H^P F\left(W^t, H_k\right) \tag{30}$$

where $\nabla_H^P F(W^t, H_k) = P(H_k - \nabla_H F(W^t, H_k)) - H_k$ is projected gradient, and λ_k is the step size determined by the Armijo rule. CBGP unnecessarily obtains the optimum for both (2) and (3) in each iteration round, but it guarantees the global convergence. This is theoretically significant. However, the Armijo-type line search in PG, PBB, and CBGP are time-consuming.

In contrast to the aforementioned gradient-based solvers, Kim and Park [16] generalized AS [16] to solve (2) and (3), which exchanges multiple variables between *free set* and *active set*, and proposed block principal pivoting method (BPP, [15]). The variable exchanging rule is

$$F = (F - C_1) \cup C_2, G = (G - C_2) \cup C_1 \tag{31}$$

where F and G are *free set* and *active set*, respectively. In (31), C_1 and C_2 are calculated by $C_1 = \{i \in F : \vec{h}_i < 0\}$ and $C_2 = \{i \in G : \vec{y}_i < 0\}$, respectively, wherein $\vec{y}$ is the Lagrange multiplier for $\vec{h}$ that is a single column of the matrix H. Both AS and BPP assume (2) and (3) are strictly convex to make the unconstrained equation with respect to variables in *free set* solvable. Such assumption may end up with numerical instability problem especially for large scale problems.

The proposed NeNMF solver converges at the optimal rate $O\left(\frac{1}{k^2}\right)$ in each iteration round without time-consuming line search procedure and predefined tuning parameters. Compared to AS and BPP, NeNMF overcomes the numerical instability problem. Therefore, NeNMF is more efficient and effective than the aforementioned NMF solvers.

III. NeNMF for Regularized NMF

This section shows that NeNMF can be conveniently adopted for optimizing the L_1-norm, L_2-norm and manifold regularized NMF.

A. NeNMF for L_1-Norm Regularized NMF

Although the final matrix factors obtained by NMF are sparse, their sparsities are not explicitly guaranteed. Hoyer [13] proposed nonnegative sparse coding (NSC) that incorporates the L_1-norm regularization on the encoding matrix H to control its sparsity. The objective function is

$$\min_{W\geq 0, H\geq 0} \frac{1}{2}\|X - WH\|_F^2 + \beta\|H\|_1 \tag{32}$$

where $\|\cdot\|_1$ is the L_1-norm, and β is the tradeoff parameter that balance the sparsity of H and the approximation error. Since all entries in H are nonnegative, we have $\|H\|_1 = \Sigma_{ij}H_{ij}$. In [13], MUR is used for optimizing (32).

Here we show that NeNMF can be naturally adopted to efficiently solve (32). The factor W can be obtained by *Algorithm* 1 directly and we show the optimization for H as following

$$\min_{H\geq 0} r(H) = \frac{1}{2}\|X - WH\|_F^2 + \beta\|H\|_1. \tag{33}$$

By considering *Lemma* 1 and the convexity of $\|H\|_1$, we can show that $r(H)$ in (33) is convex. Furthermore, for any two matrices $H_1, H_2 \in \mathbf{R}^{r\times n}$, we have

$$\begin{aligned}\|\nabla r(H_1) - \nabla r(H_2)\|_F &= \left\|W^TW(H_1 - H_2)\right\|_F\\ &\leq \|W^TW\|_2 \times \|H_1 - H_2\|_F,\end{aligned}$$

where $\nabla r(H)$ is the gradient of $r(H)$, i.e., $\nabla r(H) = W^TWH - W^TX + \beta E$, wherein $E \in \mathbf{R}^{r\times n}$ is the matrix whose entries are all one, and $\|Z\|_2$ is the matrix spectral norm, i.e., the largest singular value of Z. Thus $\nabla r(H)$ is Lipschitz continuous and the Lipschitz constant $L^{L_1} = \|W^TW\|_2$.

According to [23], (33) can be solved by OGM. We slightly modify NeNMF by replacing the gradient and the Lipschitz constant in *Algorithm* 2 with $\nabla r(H) = W^TWH - W^TX + \beta E$ and $L^{L_1} = \|W^TW\|_2$, respectively, to learn the L_1-norm regularized NMF. We term the modified NeNMF for the L_1-norm regularized NMF as NeNMF-L_1. Note that NeNMF can be used to handle other L_1-norm-based regularizations, e.g., sparse transfer learning (STL) [29].

B. NeNMF for L_2-Norm Regularized NMF

The L_2-norm regularization, i.e., Tikhonov regularization, is usually utilized to control the solution smoothness in NMF [26], i.e.

$$\min_{W\geq 0, H\geq 0} \frac{1}{2}\|X - WH\|_F^2 + \frac{\alpha}{2}\|W\|_F^2 + \frac{\beta}{2}\|H\|_F^2 \tag{34}$$

For optimizing (34), a gradient descent with constrained least squares (GD-CLS) solver is developed in [26]. GD-CLS updates W by using MUR and updates H by solving a NLS. It has been shown that BFGS [17] can be extended to solve (34). We denote this solver as BFGS-L_2.

We show that (34) can be efficiently solved by slightly modifying NeNMF. Since the optimization of H and that of W are symmetric, we only show the optimization of H. Given W, the objective function for optimizing H is

$$\min_{H\geq 0} g(H) = \frac{1}{2}\|X - WH\|_F^2 + \frac{\beta}{2}\|H\|_F^2. \tag{35}$$

By considering *Lemma* 1 and the convexity of $\|H\|_F^2$, $g(H)$ is convex. Furthermore, for any $H_1, H_2 \in \mathbf{R}_+^{r\times n}$, we have

$$\begin{aligned}\|\nabla g(H_1) - \nabla g(H_2)\|_F &= \left\|\left(W^TW + \beta I_r\right)(H_1 - H_2)\right\|_F\\ &\leq \|W^TW + \beta I_r\|_2 \times \|H_1 - H_2\|_F,\end{aligned}$$

where $\nabla g(H) = W^TWH - W^TX + \beta H$ is the gradient of $g(H)$ and $I_r \in \mathbf{R}^{r\times r}$ is the identity matrix. Thus $\nabla g(H)$ is Lipschitz continuous, and the Lipschitz constant is the spectral norm of $W^TW + \beta I_r$, i.e., $L^{L_2} = \|W^TW + \beta I_r\|_2$. Since the problem (35) is convex, it can be solved by OGM. Thus, it is easy to extend NeNMF for optimizing (34) by slightly modifying the gradient and the step size, i.e., L^{L_2}. In this paper, we name the modified NeNMF for L_2-norm regularized NMF as NeNMF-L_2.

C. NeNMF for Manifold Regularized NMF

Manifold regularization is utilized to encode the intrinsic geometric structure of data. Cai *et al.* [5] proposed the graph regularized NMF (GNMF)

$$\min_{W\geq 0, H\geq 0} \frac{1}{2}\|X - WH\|_F^2 + \frac{\gamma}{2}tr\left(HL_gH^t\right) \tag{36}$$

where L_g is the graph Laplacian of the data matrix X. It is defined by $L_g = D - S$, wherein S is the similarity matrix in the adjacent graph and an entry in the diagonal matrix $D_{ii} = \sum_j S_{ij}$. To optimize H, MUR is used in GNMF based on the following update rule:

$$H \leftarrow H \otimes \frac{W^TX + HS}{W^TWH + HD}. \tag{37}$$

In this paper, we use NeNMF to solve GNMF. It is clear that W can be optimized by using *Algorithm* 1. Here we focus on the optimization of H, and the corresponding objective function is

$$\min_{H\geq 0} \varphi(H) = \frac{1}{2}\|X - WH\|_F^2 + \frac{\gamma}{2}tr\left(HL_gH^T\right). \tag{38}$$

The gradient of $\varphi(H)$ is $\nabla\varphi(H) = W^TWH - W^TX + \gamma HL_g$. In order to use NeNMF to solve (38), we need the Lipschitz constant of $\nabla\varphi(H)$ to determine the step size. Since $\varphi(H)$ is a linear combination of $F(W,H) = \frac{1}{2}\|X - WH\|_F^2$ and $\mu(H) = tr(HL_gH^T)$, the Lipschitz constant of $\nabla\varphi(H)$ can be calculated as a linear combination of the Lipschitz constants of the $\nabla_H F(W,H)$ and $\nabla\mu(H)$, wherein $\nabla\mu(H) = HL_g$ is the gradient of $\mu(H)$. According to the following *Proposition* 2 and *Lemma* 2, we find that $\nabla\varphi(H)$ is Lipschitz continuous and the Lipschitz constant is $L^M = \|W^TW\|_2 + \gamma\|L_g\|_2$. By replacing L^M and $\nabla\varphi(H)$ with the step size and gradient in *Algorithm* 1, respectively, we can solve (38) by using NeNMF. In this paper, we denote the extended NeNMF for manifold regularized NMF as NeNMF-M.

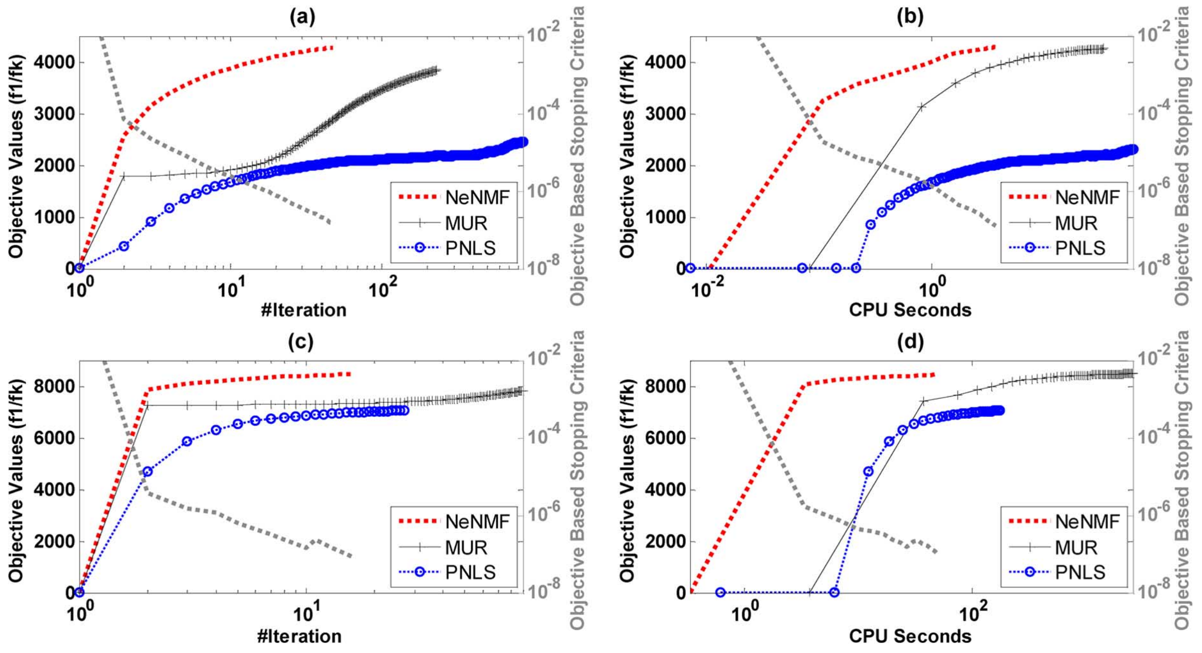


Fig. 1. Average objective values versus iteration numbers and CPU seconds of NeNMF, MUR, and PNLS on the *Synthetic 1* (a) and (b) and *Synthetic 2* (c) and (d) datasets. The iteration numbers and CPU seconds are shown in *log* scale. All solvers start from the same initial point and stop when the same stopping criterion (39) is met, wherein the precision $\tau = 10^{-7}$.

Proposition 2: The gradient of $\mu(H)$ is Lipschitz continuous and the corresponding Lipstchitz constant is $\|L_g\|_2$.

Proof: For any two matrices $H_1, H_2 \in \mathbf{R}^{r\times n}$, we have

$$\begin{aligned}
&\|\nabla\mu(H_1) - \nabla\mu(H_2)\|_F^2 \\
&\quad = \|(H_1 - H_2)L_g\|_F^2 \\
&\quad = tr\left(((H_1 - H_2)L_g)^T((H_1 - H_2)L_g)\right) \\
&\quad = tr\left(((H_1 - H_2)U\Sigma U^T)\left((H_1 - H_2)U\Sigma U^T\right)^T\right) \\
&\quad = tr\left(U^T(H_1 - H_2)^T(H_1 - H_2)U\Sigma^2\right) \\
&\quad \le \delta_1^2 tr\left(U^T(H_1 - H_2)^T(H_1 - H_2)U\right)
\end{aligned}$$

where $U\Sigma U^T$ is the SVD decomposition of L_g and the singular values are $\{\delta_1, \ldots, \delta_v\}$ arranged in a descending order. Since $UU^T = I_n$, wherein $I_n \in \mathbf{R}^{n\times n}$ is the identity matrix, we have

$$\|\nabla\mu(H_1) - \nabla\mu(H_2)\|_F \le \|L_g\|_2 \times \|H_1 - H_2\|_F$$

where $\|L_g\|_2 = \delta_1$ is the spectral norm of L_g. Therefore, $\nabla\mu(H)$ is Lipschitz continuous and the Lipstchitz constant is $\|L_g\|_2$. ■

Since NeNMF-M does not divide L_g into two parts as in (37), it can be adopted to wider range of criterions such as geometric mean [8] and patch alignment framework [28], [30], [32]. Note that NeNMF-M decreases the objective function, but it cannot guarantee the optimality of the obtained solution H_K in each iteration round. That is because the objective function $\varphi(H)$ is nonconvex. Therefore, similar to MUR for GNMF, NeNMF-M cannot guarantee the convergence to any stationary point. However, experimental results in the following section show that NeNMF-M is efficient and effective for optimizing (36).

IV. Experiments

In this section, we evaluate NeNMF by comparing to following nine NMF solvers in term of efficiency and approximation accuracy:

1) Multiplicative Update Rule (MUR, [18])
2) Projected Nonnegative Least Squares (PNLS,[2] [2])
3) Projected Gradient (PG,[3] [20])
4) Projected Quasi-Newton (QN,[4] [33])
5) Broyden Fletcher Goldfarb Shanno (BFGS,[5] [17])
6) Projected Barzilai Borwein (PBB,[6] [12])
7) Cyclic Block Coordinate Gradient Projection (CBGP, [4])
8) Active Set (AS,[7] [16])
9) Block Principal Pivoting (BPP,[8] [15])

The source codes for PNLS, PG, BFGS, AS, BPP, and PBB are available online.[2345678] The source code of QN is given in NMFLAB toolbox.[4] It is implemented to optimize the Frobenius norm-based nonnegative matrix factorization (1) by using the Quasi-Newton update (26), whereas the parameter λ starts from 10^{-13} and increases until the regularized Hessian matrix is ill-conditioned. We implemented other algorithms in MATLAB. Since the implementation of QN cannot handle large size matrices, we only presented its results on small size matrices. In the first part, we compared NeNMF to MUR and

[2]The code was found at http://www.cs.utexas.edu/users/dmkim/Source/software/nnma/index.html

[3]The code is available at http://www.csie.ntu.edu.tw/cjlin/nmf/index.html

[4]The NMFLAB toolbox is available at http://www.bsp.brain.riken.go.jp/ICALAB/nmflab.html

[5]The code is available at http://www.cs.utexas.edu/users/dmkim/Source/software/nnma/index.html

[6]The code is available at http://www.math.uconn.edu/neumann/nmf/

[7]The code is available at http://www.cc.gatech.edu/hpark/nmfsoftware.php

[8]The code is available at http://www.cc.gatech.edu/hpark/nmfsoftware.php

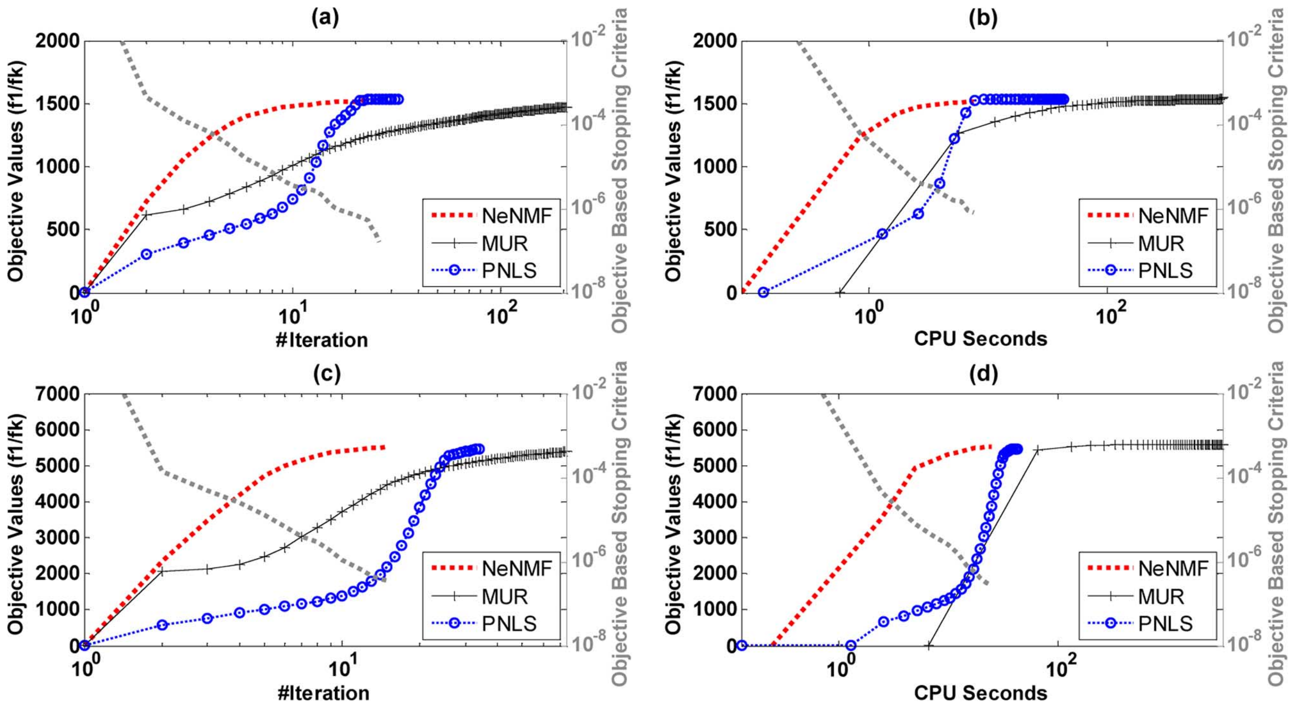


Fig. 2. Average objective values versus iteration numbers and CPU seconds of NeNMF, MUR , and PNLS on the *Reuters-21578* (a) and (b) and *TDT-2* (c) and (d) datasets. The iteration numbers and CPU seconds are shown in *log* scale. All solvers start from the same initial point and stop when the same stopping criterion (39) is met, wherein the precision $\tau = 10^{-7}$.

TABLE II
PROBLEM SUMMARY OF SYNTHETIC AND REAL-WORLD DATASETS

Dataset	m	n	r
Synthetic 1	500	100	50
Synthetic 2	5,000	1,000	100
Reuters-21578	1,893	829	50
TDT-2	3,677	939	100

PNLS-based NMF solvers. In the second part, we compared NeNMF to PG and AS-based NMF solvers. All the NMF solvers were compared in terms of their efficiencies on both synthetic and real-world datasets summarized in Table II. The synthetic datasets include 500 $\times$ 100-dimension and 5,000 $\times$ 1,000-dimension random dense matrices and the real-world datasets include Reuters-21578[9] and TDT-2[10] datasets. The Reuters-21578 dataset contains 8,293 documents represented by an 18,933 $\times$ 8,293-dimension matrix, and the TDT-2 dataset contains 9,394 documents represented by a 36,771 $\times$ 9,394-dimension matrix. The following Section IV-C will introduce these two datasets in detail. For each datasets, around 1/10 columns and rows were randomly selected to form a submatrix whose size is depicted in Table II. Different low rank r and matrices size (m, n) were set to evaluate the scalability of NeNMF and the initial points of all solvers were set identical for fair comparison.

[9]Reuters-21578 corpus is available at http://www.daviddlewis.com/resources/testcollections/reuters21578

[10]Nist Topic Detection and Tracking corpus is available at http://www.itl.nist.gov/iad/mig/tests/tdt/2001/dryrun.html

According to Section II, NeNMF obtains a better approximation of the original matrix with the same tolerance than existing NMF solvers. To evaluate the effectiveness of the solution obtained by NeNMF, we compared the document clustering results on two popular datasets, i.e., Reuters-21578 [19] and TDT-2 [7], with other NMF solvers.

A. NeNMF Versus MUR and PNLS

Although the stopping criterion defined in (22) works well for checking the stationarity of limit points, it is not suitable for MUR [18] and PNLS [2]. That is because these solvers do not guarantee the convergence to any stationary point. A usual stopping criterion for MUR and PNLS is

$$\frac{F(W^t, H^t) - F(W^*, H^*)}{F(W^1, H^1) - F(W^*, H^*)} \leq \tau \tag{39}$$

where τ is a predefined precision, and (W^*, H^*) is the final solution. Since it is usually impossible to know (W^*, H^*) in advance, we use (W^{t+1}, H^{t+1}) instead. For fair comparison, we replaced the stopping criterion (22) with (39) in NeNMF in this experiment. The precision is empirically set to a small value, e.g., $\tau = 10^{-7}$. This strategy for setting τ is standard in optimization.

Note that the stopping criterion (39) needs the calculation of the objective function $F(W, H)$ at each iteration round. It is time-consuming, similar to [20], so we reformulate (1) to reduce the time cost of (39)

$$\begin{aligned}\|X - WH\|_F^2 \\ &= tr(X^T X) - 2tr(H^T W^T X) + tr(W^T W H H^T).\end{aligned} \tag{40}$$

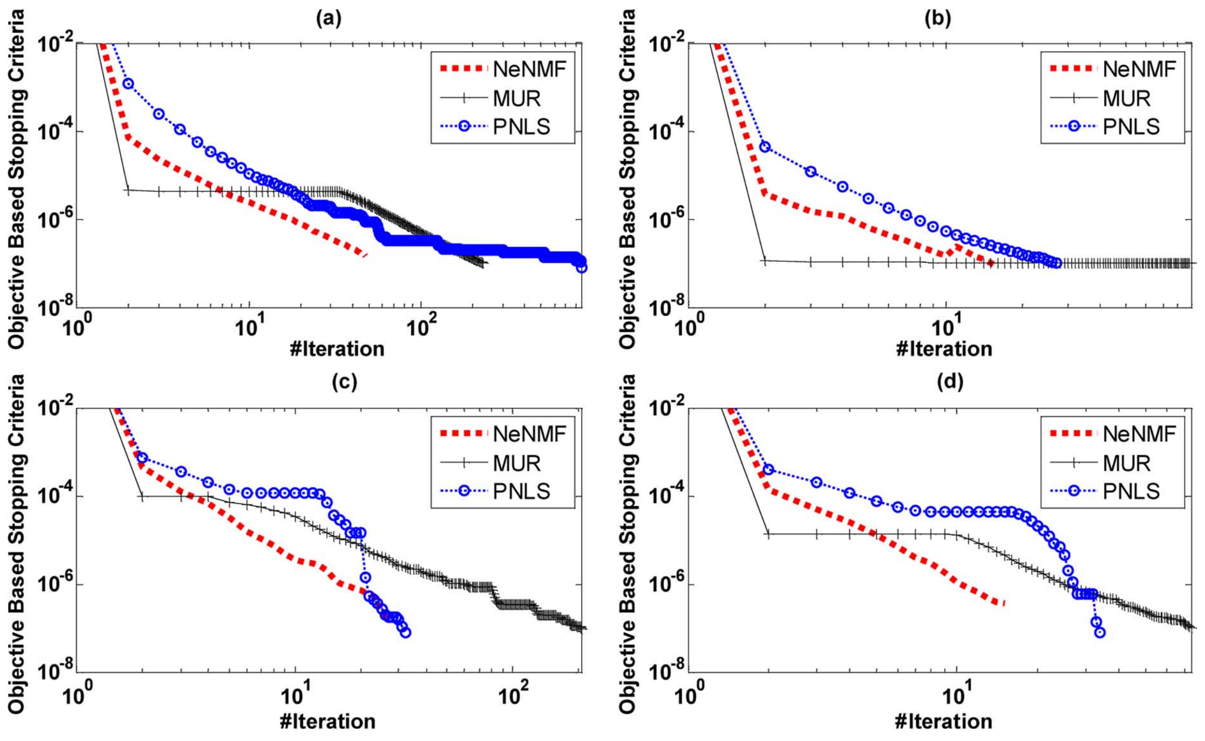


Fig. 3. Average objective-based stopping criteria values versus iteration numbers of NeNMF, MUR and PNLS on *Synthetic 1* (a), *Synthetic 2* (b), *Reuter-21578* (c), and *TDT-2* (d) datasets. The objective-based stopping criteria values are shown in *log* scale. All solvers start from the same initial point and stop when the same stopping criterion (39) is met, wherein the precision $\tau = 10^{-7}$.

Since the first term $tr(X^TX)$ in (40) can be calculated as $\|X\|_F^2$ before iterations of NeNMF, the second and third terms can be calculated as

$$tr(H^TW^TX) = \sum_{ij} h_{ij}(W^TX)_{ij}$$

and

$$tr(W^TWHH^T) = \sum_{ij}(W^TW)_{ij}(HH^T)_{ij}$$

where W^TX and W^TW have been calculated in the *Algorithm* 1, the main time cost of (40) is spent on calculating HH^T whose time complexity is $O(nr^2)$. Based on the trick (40), we can reduce the complexity of $F(W,H)$ from $O(mnr)$ to $O(nr^2)$, because $r \ll \min\{m,n\}$. The same trick was used to the other NMF solvers for fair comparison.

We compared NeNMF to MUR and PNLS on the *Synthetic 1* and *Synthetic 2* datasets in terms of efficiency. All the solvers start from the same initial point (W^1, H^1), where both W^1 and H^1 are random dense matrices, and stop when (39) is satisfied, where the precision $\tau = 10^{-7}$. We conducted the experiment ten trials. Fig. 1 shows the average objective values versus iteration numbers and CPU seconds. According to this figure, NeNMF reduces the objective function much faster than MUR and PNLS in same iteration rounds. That is because each iteration round of NeNMF minimizes one matrix factor (W or H) with another fixed at the convergence rate of $O\left(\frac{1}{k^2}\right)$ while neither MUR nor PNLS does that. Although NeNMF is comparable to MUR on the *Synthetic 1* dataset [cf. Fig. 1(a) and (b)] in terms of CPU seconds, it spends much less CPU seconds than MUR on the *Synthetic 2* dataset [see Fig. 1(c) and (d)]. That is because the time complexity of NeNMF is comparable to that of MUR when $m \gg r$ and $n \gg r$ (see Table I), but it reduces the objective function much lower than MUR in each iteration round. Fig. 1(a) and (b) show that PNLS suffers from the nonconvergence problem caused by the numerical instability of pseudoinverse operator in (24), i.e., the matrix HH^T will be of rank-deficient when $m = 500$, $n = 100$, and $r = 50$.

We further compared their efficiencies on real-world datasets, *Reuters-21578* and *TDT-2*. All solvers start from the same randomly generated initial point and stop when (39) is satisfied ($\tau = 10^{-7}$). We conducted the experiments 10 trials. Fig. 2 shows the average objective values versus iteration numbers and CPU seconds.

The results shown in Fig. 2 are consistent with those shown in Fig. 1. In Figs. 1 and 2, the gray scale right-hand side (RHS) axis is the objective-based stopping criteria of NeNMF, i.e.

$$\frac{F(W^t,H^t) - F(W^*,H^*)}{F(W^1,H^1) - F(W^*,H^*)}$$

defined in (39). It shows that NeNMF converges rapidly on both synthetic and real-world datasets.

Fig. 3 shows the average objective-based stopping criteria defined in(39) versus iteration numbers of NeNMF, MUR, and PNLS on the *Synthetic 1*, *Synthetic 2*, *Reuter-21578*, and *TDT-2* datasets. Fig. 3(a) and (b) shows that NeNMF converges in less iteration than both MUR and PNLS on synthetic datasets. From Fig. 3(c) and (d), we have the same observation as that obtained from Fig. 3(a) and (b).

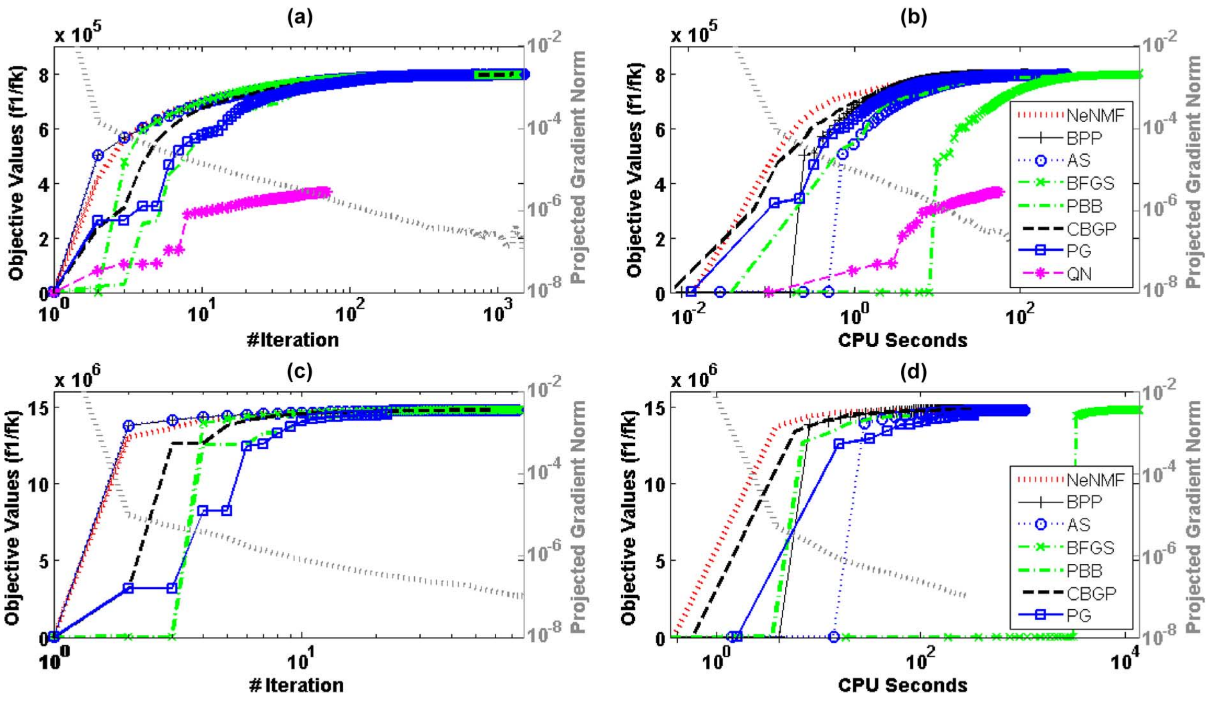


Fig. 4. Average objective values versus iteration numbers and CPU seconds of NeNMF-, PG-, and AS-based NMF solvers on *Synthetic 1* (a) and (b) and *Synthetic 2* (c) and (d) datasets. The iteration numbers and CPU seconds are given in *log* scale. All solvers start from the same initial point and stop when the same stopping criterion (22) is met, wherein the tolerance $\epsilon = 10^{-7}$.

B. NeNMF Versus PG- and AS-Based NMF Solvers

We compared NeNMF to PG- and AS-based NMF solvers in terms of efficiency on the *Synthetic 1* and *Synthetic 2* datasets. All the solvers start from the same initial point (W^1, H^1), where both W^1 and H^1 are random dense matrices, and stop when the same stopping criterion (22) is satisfied, where the tolerance $\epsilon = 10^{-7}$. We conducted the experiments ten trials. Fig. 4 shows the average objective values versus iteration numbers and CPU seconds. In this experiment, QN was only conducted on the *Synthetic 1* dataset because the implementation of QN in NMFLAB fails on large size matrices due to the numerical instability.

Fig. 4 shows that NeNMF converges in less iterations and CPU seconds than PBB, CBGP, PG, and QN on both datasets. That is because NeNMF minimizes each matrix factor (W or H) optimally in each iteration round without time-consuming line search and complex Hessian update procedures. Although the number of iterations of BFGS is comparable to that of NeNMF [cf. Fig. 4(a) and (c)], BFGS spent more CPU seconds than NeNMF [cf. Fig. 4(b) and (d)]. That is because BFGS minimizes rows of W and columns of H separately. In addition, Fig. 4 shows that NeNMF, BPP and AS share similar iteration numbers, while NeNMF requires less CPU seconds than AS and BPP.

We also compared these solvers on two real-world datasets, *Reuters-21578* and *TDT-2*. We conduct the experiment ten trails. In each trail, all solvers start from the same random dense matrices (W^1, H^1) and stop when the stopping criterion (22) is satisfied, where the tolerance $\epsilon = 10^{-7}$. Fig. 5 shows the average objective values versus iteration numbers and CPU seconds.

Results shown in Fig. 5 are consistent with those shown in Fig. 4. In Figs. 4 and 5, the gray scale RHS axis is the projected gradient norm of NeNMF, i.e.

$$\frac{\left\|\left[\nabla_H^P F(W^t, H^t), \nabla_W^P F(W^t, H^t)^T\right]\right\|_F}{\left\|\left[\nabla_H^P F(W^1, H^1), \nabla_W^P F(W^1, H^1)^T\right]\right\|_F}$$

which is defined in (22). It shows that NeNMF converges rapidly on both synthetic and real-world datasets.

Fig. 6 shows the average projected gradient norm defined in (22) versus iteration numbers of NeNMF-, PG-, and AS-based NMF solvers on *Synthetic 1*, *Synthetic 2*, *Reuter-21578*, and *TDT-2* datasets. Fig. 6(a) and (b) shows that NeNMF converges more iterations than PG, PBB, and CBGP on synthetic datasets, but it costs less CPU seconds in each iteration. Thus NeNMF performs more efficiently than PG, PBB, and CBGP [see Fig. 4(a) and (b)]. Fig. 6(c) and (d) shows that NeNMF converges in less iterations than all the other NMF solvers on real-world datasets. In addition, NeNMF avoids the complex line search procedure and thus costs less CPU seconds in each iteration round. In summary, NeNMF converges faster than all the other NMF solvers on real-world datasets.

In summary, NeNMF benefits from the optimal convergence property of OGM (cf. *Proposition* 1). It outperforms PG-based NMF solvers in terms of efficiency. Compared to AS-based NMF solvers, NeNMF performs slightly superior to BPP in terms of efficiency on both synthetic and real-world dataset. But BPP has the numerical instability problem caused by the unconstrained equation solution when the matrix HH^T is rank-deficient. In addition, NeNMF performs well for solving manifold regularized NMF while BPP cannot solve this problem because

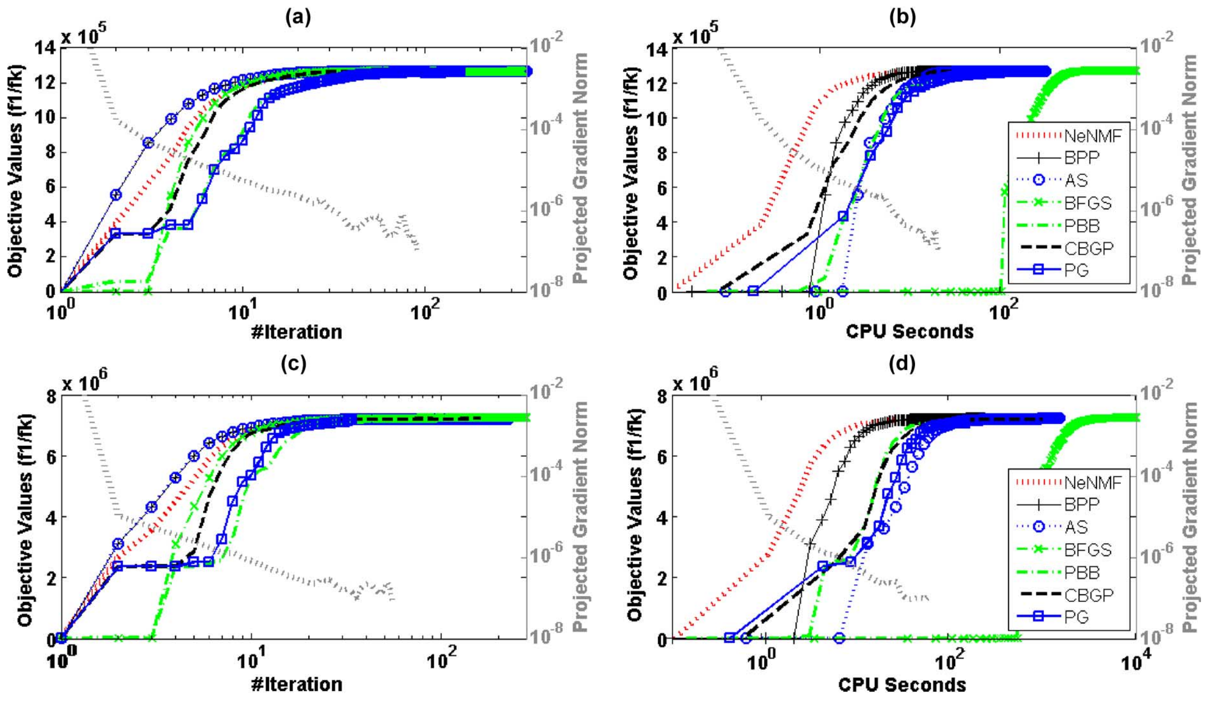


Fig. 5. Average objective values versus iteration numbers and CPU seconds of NeNMF-, PG-, and AS-based NMF solvers on the *Reuters-21578* (a) and (b) and *TDT-2* (c) and (d) datasets. The iteration numbers and CPU seconds are shown in *log* scale. All solvers start from the same initial point and stop when the same stopping criterion (22) is met, wherein the tolerance $\epsilon = 10^{-7}$.

there does not exist analytical solution for the unconstrained equation on *free set*.

C. Document Clustering on Real-World Datasets

Recently, NMF has been widely applied to document clustering [26], [31]. This section evaluates the effectiveness of NeNMF by comparing to existing NMF solvers in term of document clustering performance. Since MUR does not guarantee the stationarity of the generated limit point, we used (39) as its stopping criterion. PNLS and QN usually suffer from the nonconvergence and numerical instability problem, respectively, on large size matrices (see Section IV-A and IV-B), so they are excluded from this experiment.

In the experiment, K-means was used as a baseline. Both NMF solvers and K-means were conducted on two document datasets, Reuter-21578 [19] and TDT-2 [7]. The Reuter-21578 corpus contains 21,578 documents in 135 categories. By eliminating those documents with multiple category labels and using the largest 30 categories, we selected 8,292 documents for evaluation. The TDT-2 corpus consists of 11,201 documents collected from six news agencies includes ABC, CNN, VOA, NYT, PRI, and APW. All the documents were grouped into 96 semantic categories, each of which reports a major news event occurred during the first six months of 1998. By removing those documents appearing in two or more categories and using the largest 30 categories, we selected 9,394 documents for evaluation. Each document was represented with a normalized term-frequency vector. For both datasets, 2 to 10 categories were randomly selected for test. We conducted the experiments 50 trials, and the average accuracy [31] and the average mutual information [5] were reported in Fig. 7.

Fig. 7 shows that NeNMF outperforms K-Means, PG, and PBB on both datasets. The stopping criterion (22) checks whether the final solution obtained by a particular NMF solver is sufficiently close to a stationary point. In particular, given a positive tolerance ϵ, stopping criterion $\|[\nabla_H^P F(W^t, H^t), \nabla_W^P F(W^t, H^t)^T]\|_F \leq \epsilon \|[\nabla_H^P F(W^1, H^1), \nabla_W^P F(W^1, H^1)^T]\|_F$ measures the distance between the current solution obtained by an NMF solver and the corresponding stationary point. Given this stopping criterion, an NMF solver will stop searching a better solution when the tolerance is reached. Since both NeNMF- and PG-based NMF solvers (e.g., PG [20], PBB [12], and CBGP [4]) alternatively update the matrix factors by solving the corresponding NLS subproblems, their efficiencies mainly depend on the convergence rate of the optimization method for each NLS subproblem. In this paper, we have proved that OGM used in NeNMF converges optimally at the rate of $O\left(\frac{1}{k^2}\right)$ (see *Proposition* 1) while the NLS optimization method used in other PG-based NMF solvers (including PG [20], PBB [12], and CBGP [4]) cannot guarantee this convergence rate. Therefore, NeNMF gets closer than other PG-based NMF solvers to the stationary point in each iteration round and obtains the lowest final objective value given a same nonzero tolerance and a same initialization (see Fig. 5). However, given the zero tolerance value, it is very possible that NeNMF and other PG-based NMF solvers can reach the same objective value. Although NeNMF performs comparably to BFGS, CBGP, and MUR, it is more efficient than them as shown in Figs. 4 and 5. Fig. 7 shows NeNMF performs comparably to AS and BPP, and NeNMF overcomes their numerical instability problem (cf. Section IV-B) and performs more robustly in practice.

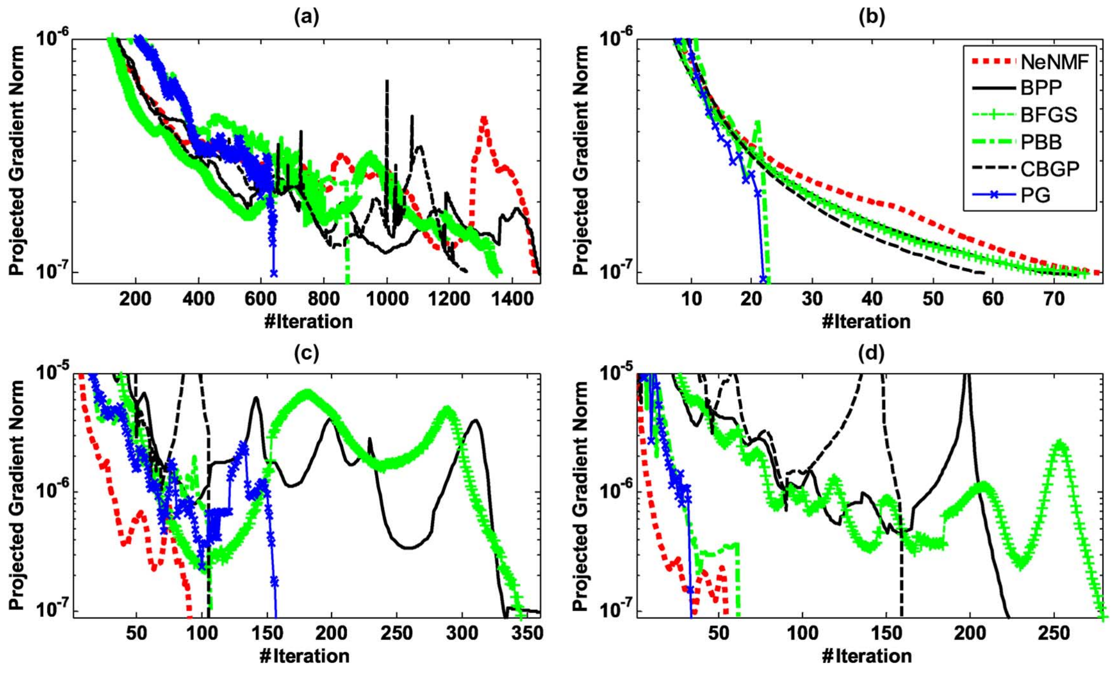


Fig. 6. Average projected gradient norm versus iteration numbers of NeNMF-, PG-, and AS-based NMF solvers on *Synthetic 1* (a), *Synthetic 2* (b), *Reuter-21578* (c), and *TDT-2* (d) datasets. The projected gradient norm is shown in *log* scale. All solvers start from the same initial point and stop when the same stopping criterion (22) is met, wherein the tolerance $\epsilon = 10^{-7}$.

In the above experiment, different NMF solvers performed very differently because the used stopping criterion (22) checks whether the final solution obtained by a particular NMF solver is close to a stationary point. This criterion makes an algorithm stop at a certain stationary point whose approximation error can be large. Therefore, some algorithms stopped by the stopping criterion (22) performed poorly.

Although the aforementioned evaluations are important to demonstrate how different NMF solvers perform based on this stopping criterion (22), it could be unfair to show the performance evaluation based only upon this criterion. Therefore, we further evaluate the effectiveness of the objective values obtained by different NMF solvers to the clustering performance.

We repeated the clustering experiments, whereas both NeNMF- and PG-based NMF solvers stop when an identical objective value was reached. In this experiment, the objective value was set to the final objective value of NeNMF when the stopping criterion (22) is met with tolerance $\epsilon = 10^{-7}$. Fig. 8(a), (b), (d), and (e) shows that all NMF solvers perform comparably when an identical objective value is reached on both datasets. Fig. 8(c) and (f) shows that NeNMF costs less CPU seconds than other NMF solvers.

D. Efficiency Evaluation of NeNMF Extensions

In Section III, we show that NeNMF can be easily extended for L_1-norm, L_2-norm and manifold regularized NMF. This section evaluates the efficiencies of these NeNMF extensions. Since NSC [13], GD-CLS [26] and GNMF [5] are MUR-based solvers, we choose (39) as the stopping criterion for fair comparison.

To evaluate the efficiency of NeNMF-L_1, we compared its objective values to NSC [13] on a 1,600 $\times$ 320-dimension random dense matrix in Fig. 9. In this experiment, we set the tradeoff parameter β in (32) to 1,000 for both solvers and conducted them from the same initial point. For NSC, the stepsize was set to a small value 0.01 to ensure its convergence. Fig. 9 shows that NeNMF-L_1 further reduces the objective function in less iterations and CPU seconds than NSC. Hence, NeNMF-L_1 performs more efficiently than NSC.

To evaluate the efficiency of NeNMF-L_2, we compared its objective values to those of GD-CLS [26] and BFGS-L_2 on a $1{,}600 \times 320$-dimension random dense matrix in Fig. 10. In this experiment, we set the tradeoff parameters in (34) as $\alpha = 0.1$, $\beta = 0.1$ for all solvers. The initial points of all solvers are identical. Fig. 10(a) shows that NeNMF-L_2 reduces the objective value faster than GD-CLS because it obtains the optimal solution of both matrix factors in each iteration round while GD-CLS does not. Fig. 10(b) presents that NeNMF-L_2 solver achieves the lowest objective value among these three solvers in same CPU seconds. NeNMF-L_2 performs more efficiently than GD-CLS and BFGS-L_2 for L_2-norm regularized NMF.

To evaluate the efficiency of NeNMF-M, we compared its objective values to those of GNMF on a $1{,}600 \times 320$-dimension random dense matrix in Fig. 11. In our experiment, according to [5], the tradeoff parameter in (36) was set to $\gamma = 1{,}000$ and the number of nearest neighbor k was set to 5. The edge weight of the adjacent graph in (36) is measured by the heat kernel with the predefined parameter $\delta = 1$. The initial points of both solvers were set identical. Fig. 11 shows that NeNMF-M further reduces the objective function in less iterations and CPU seconds compared to GNMF. That is because NeNMF-M searches each matrix factor (W or H) optimally at each iteration round while GNMF searches just one step along the rescaled gradient direction.

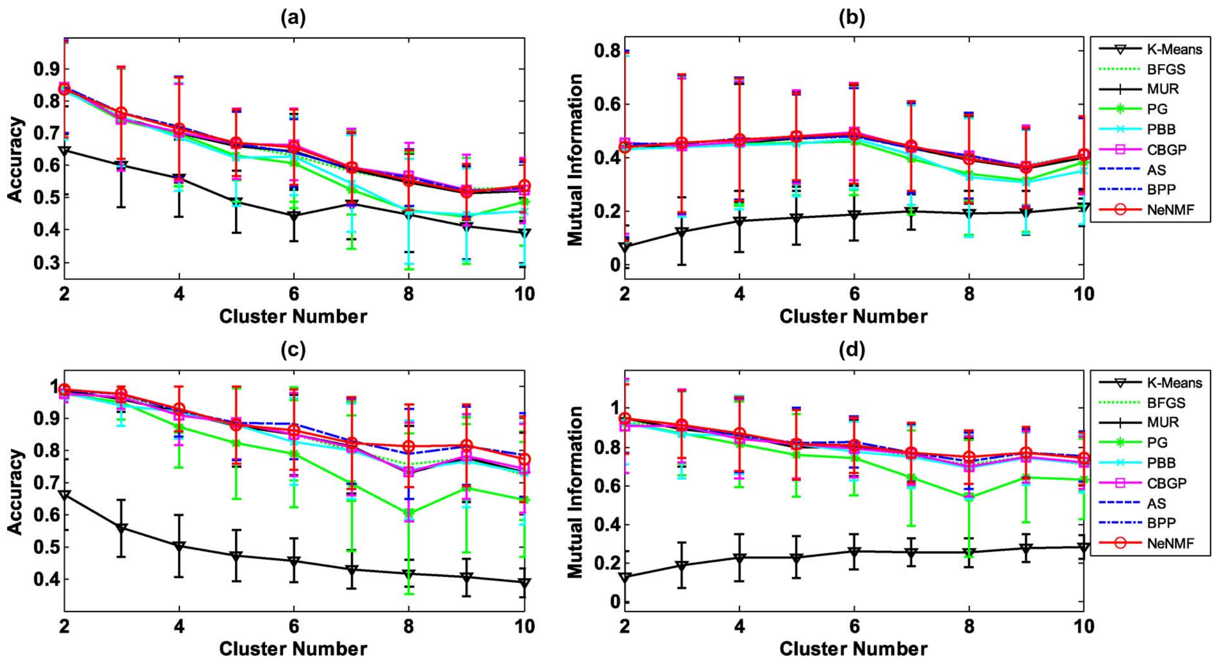


Fig. 7. The average accuracy and mutual information versus the cluster number obtained by NeNMF or other NMF solvers on Reuter-21578 (a) and (b) and TDT-2 (c) and (d) datasets. All solvers start from the same initial point and stop when the same stopping criterion (22) is met, wherein the tolerance $\epsilon = 10^{-7}$. The stopping criterion of MUR is given by (39), wherein the precision $\tau = 10^{-7}$.

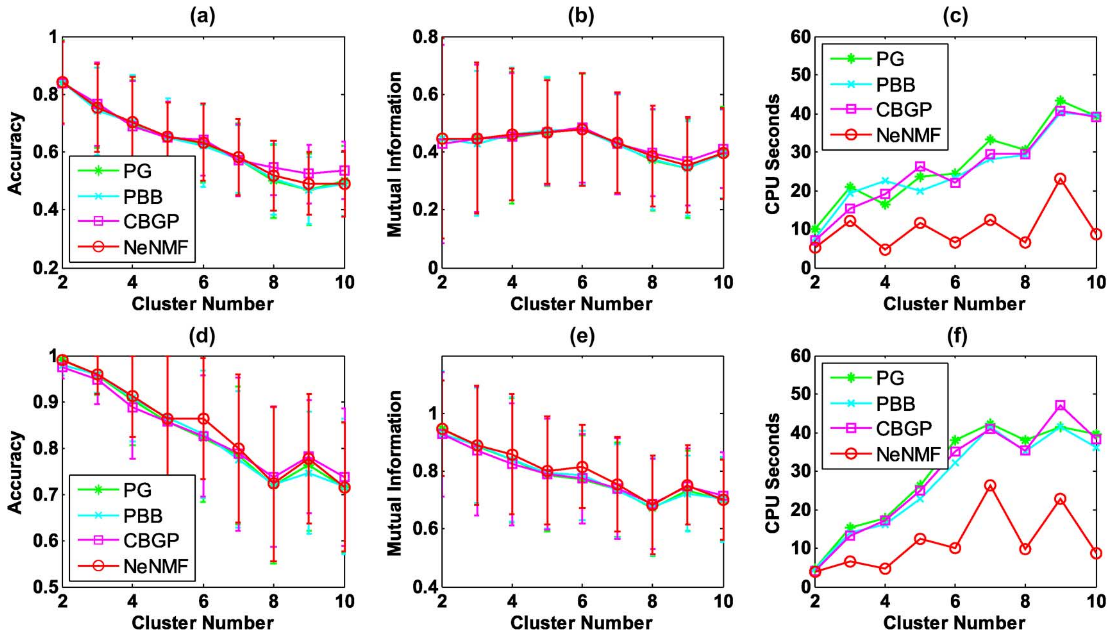


Fig. 8. The average accuracy and mutual information versus the cluster number obtained by NeNMF, PG, PBB, CBGP, AS, and BPP on Reuter-21578 (a) and (b) and TDT-2 (c) and (d) datasets. All solvers start from the same initial point and stop with identical objective value, which was set to the final objective value obtained by NeNMF. In this experiment, NeNMF stopped when the criterion (22) is met, wherein the tolerance $\epsilon = 10^{-7}$.

E. Remarks

Based on the above experiments, we have the following remarks:

1) The precision τ for the stopping criterion (39) and the tolerance ϵ for the stopping criterion (22) were set to a small value, i.e., 10^{-7}, in our experiments. We have consistent results on smaller values.
2) In our experiments, the objective values were shown as how far the final objective values are from the initial one, i.e., $\frac{F(W^1,H^1)}{F(W^k,H^k)}$. Both the iteration numbers and CPU seconds were shown in *log* scale to better show the convergence details.
3) In the clustering experiments, NeNMF outperformed most PG-based NMF solvers. That is because NeNMF obtains a better approximation, i.e., the objective value obtained by NeNMF is lower than those obtained by PG-based NMF solvers given the same tolerance, i.e., 10^{-7}. The performance of PG-based NMF solvers can be improved by re-

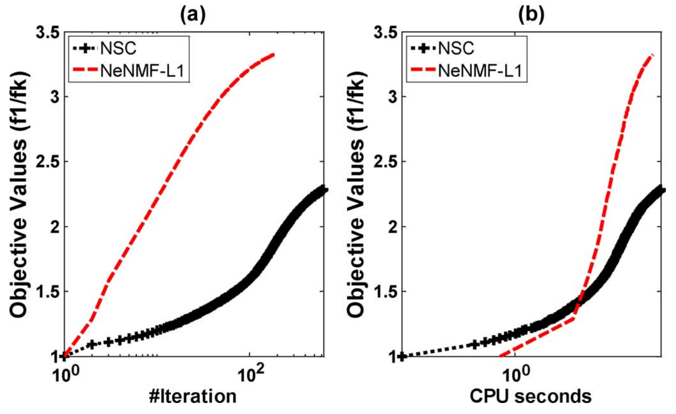


Fig. 9. The objective values versus (a) iteration number and (b) CPU seconds of NeNMF-L_1 and NSC on 1,600 × 320-dimension dense matrix, where the low rank is 64. Both solvers start from the same initial point obtained by one-step MUR on random generated matrices, and stop when the stopping criterion (39) is satisfied, wherein the precision $\tau = 10^{-4}$. Both the iteration numbers and CPU seconds are shown in *log* scale.

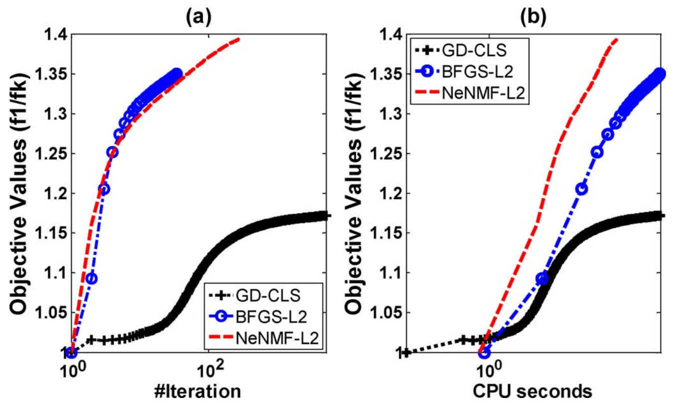


Fig. 10. The objective values versus (a) iteration numbers and (b) CPU seconds of NeNMF-L_2, GD-CLS and BFGS-L_2 on 1,600 × 320-dimension dense matrix, where the low rank is 64. All solvers start from the same initial point obtained by one-step MUR on random generated matrices, and stop when the stopping criterion (39) is satisfied, wherein the precision $\tau = 10^{-4}$. Both the iteration numbers and CPU seconds are shown in *log* scale.

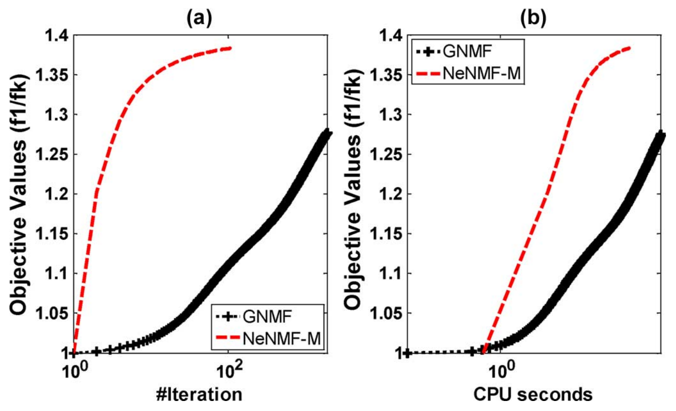


Fig. 11. The objective values versus (a) iteration number and (b) CPU seconds of NeNMF-M and GNMF on 1,600 × 320-dimension dense matrix, where the low rank is 64. Both solvers start from the same initial point obtained by one-step MUR on random generated matrices, and stop when the stopping criterion (39) is satisfied, wherein the precision $\tau = 10^{-4}$. The iteration numbers and CPU seconds are shown in *log* scale.

ducing the tolerance. We do not discuss this problem because it is beyond the scope of this paper.

V. Conclusion

This paper presents a new efficient nonnegative matrix factorization solver NeNMF, which sequentially optimizes one matrix factor with another fixed by using Nesterov's method. In NeNMF, besides the approximate solution sequence we construct another sequence which contains the search points constructed by linearly combining the latest two approximate solutions. Since the structure information is incorporated in optimization, NeNMF converges optimally at rate of $O\left(\frac{1}{k^2}\right)$ in optimizing each matrix factor with another fixed. Hence NeNMF accelerates the NMF optimization without time-consuming line search procedure or numerical instability problems. Preliminary experiments on both synthetic and real-world datasets show that NeNMF outperforms existing NMF solvers in terms of efficiency and overcomes their deficiencies. The clustering experiments on well-known real-world text datasets confirm the effectiveness of NeNMF. In addition, we show that our NeNMF solver can be naturally extended for optimizing several NMF extensions including L_1-norm, L_2-norm and manifold regularized NMF.

Appendix A
Proof of Lemma 1

Proof: Given any two matrices $H_1, H_2 \in \mathbf{R}^{r\times n}$ and a positive number $\lambda \in (0,1)$, we have

$$\begin{aligned}
&F\left(W^t, \lambda H_1 + (1-\lambda)H_2\right)\\
&\quad - \left(\lambda F(W^t, H_1) + (1-\lambda)F(W^t, H_2)\right)\\
&\qquad = \frac{1}{2}tr\Big(\left(X - W^t\left(\lambda H_1 + (1-\lambda)H_2\right)\right)^T\\
&\qquad\qquad \times \left(X - W^t\left(\lambda H_1 + (1-\lambda)H_2\right)\right)\Big)\\
&\qquad\quad - \frac{\lambda}{2}tr\left((X - W^tH_1)^T(X - W^tH_1)\right)\\
&\qquad\quad - \frac{1-\lambda}{2}tr\left((X - W^tH_2)\right)^T\left(X - W^tH_2\right)) \qquad (41)
\end{aligned}$$

where $tr(\cdot)$ is the matrix trace operator. By some algebra, (41) is equivalent to

$$\begin{aligned}
&F\left(W^t, \lambda H_1 + (1-\lambda)H_2\right)\\
&\quad - \left(\lambda F(W^t, H_1) + (1-\lambda)F(W^t, H_2)\right)\\
&\qquad = -\frac{\lambda(1-\lambda)}{2}tr\left(\left(W^t(H_1 - H_2)\right)^T\left(W^t(H_1 - H_2)\right)\right)\\
&\qquad = -\frac{\lambda(1-\lambda)}{2}\left\|W^t(H_1 - H_2)\right\|_F^2 \leq 0.
\end{aligned}$$

Therefore, we have

$$F\left(W^t, \lambda H_1+(1-\lambda)H_2\right) \leq \lambda F(W^t, H_1)+(1-\lambda)F(W^t, H_2).$$

According to the definition of convex function, we know $F(W^t, H)$ is convex. This completes the proof. ■

Appendix B
Proof of Lemma 2

Proof: According to (2), we can obtain the gradient of $F(W^t, H)$

$$\nabla_H F(W^t, H) = {W^t}^T W^t H - {W^t}^T X. \tag{42}$$

For any two matrices $H_1, H_2 \in \mathbf{R}^{r\times n}$, we have

$$\begin{aligned}&\left\|\nabla_H F(W^t, H_1) - \nabla_H F(W^t, H_2)\right\|_F^2\\ &= \left\|{W^t}^T W^t (H_1 - H_2)\right\|_F^2\\ &= tr\left(\left(U\Sigma U^T(H_1 - H_2)\right)^T \left(U\Sigma U^T(H_1 - H_2)\right)\right)\end{aligned} \tag{43}$$

where $U\Sigma U^T$ is the SVD decomposition of ${W^t}^T W^t$ and the singular values are $\{\delta_1, \ldots, \delta_u\}$ arranged in a descending order. By some algebra, (43) is equivalent to

$$\begin{aligned}&\left\|\nabla_H F(W^t, H_1) - \nabla_H F(W^t, H_2)\right\|_F^2\\ &= tr\left(U^T(H_1 - H_2)(H_1 - H_2)^T U\Sigma^2\right)\\ &\leq \delta_1^2 tr\left(U^T(H_1 - H_2)(H_1 - H_2)^T U\right)\\ &= \delta_1^2 \|H_1 - H_2\|_F^2\end{aligned} \tag{44}$$

where δ_1 is the largest singular value, and the last two equations come from the fact that $U^T U = I_u$ and $UU^T = I_r$, wherein both $I_u \in \mathbf{R}^{u\times u}$ and $I_r \in \mathbf{R}^{r\times r}$ are identity matrices. From (44), we have

$$\left\|\nabla_H F(W^t, H_1) - \nabla_H F(W^t, H_2)\right\|_F \leq L\|H_1 - H_2\|_F.$$

Therefore, $\nabla_H F(W^t, H)$ is Lipschitz continuous and the Lipschitz constant is the largest singular value of ${W^t}^T W^t$, i.e., $L = \|{W^t}^T W^t\|_2$. This completes the proof. ■

Appendix C
Proof of Proposition 1

Proof: According to **Theorem 2.2.7** in [23], for any $H \in \mathbf{R}_+^{r\times n}$ and $Y \in \mathbf{R}^{r\times n}$, we have

$$\begin{aligned}F(W^t, H) \geq F\left(W^t, P_L(Y)\right) &+ L\langle P_L(Y) - Y, Y - H\rangle\\ &+\frac{L}{2}\left\|Y - P_L(Y)\right\|_F^2\end{aligned} \tag{45}$$

where $P_L(Y) = \arg\min_{X\geq 0}\phi(Y, X)$. By substituting $H = H_k, Y = Y_{k+1}$ and $H = H_*, Y = Y_{k+1}$ into (45) and using the equality $H_{k+1} = P_L(Y_{k+1})$ derived from (15), we have

$$\begin{aligned}F(W^t, H_k) \geq F(W^t, H_{k+1}) &+ \frac{L}{2}\|Y_{k+1} - H_{k+1}\|_F^2\\ &+L\langle H_{k+1} - Y_{k+1}, Y_{k+1} - H_k\rangle\end{aligned} \tag{46}$$

and

$$\begin{aligned}F(W^t, H_*) \geq F(W^t, H_{k+1}) &+ \frac{L}{2}\|Y_{k+1} - H_{k+1}\|_F^2\\ &+L\langle H_{k+1} - Y_{k+1}, Y_{k+1} - H_*\rangle.\end{aligned} \tag{47}$$

Since $\alpha_{k+1} > 1$, by multiplying both sides of (46) by $\alpha_{k+1} - 1$ and adding it to (47), we have

$$\begin{aligned}&(\alpha_{k+1} - 1)F(W^t, H_k) + F(W^t, H_*)\\ &\quad\geq \alpha_{k+1}F(W^t, H_{k+1})\\ &\qquad + L\langle H_{k+1} - Y_{k+1}, \alpha_{k+1}Y_{k+1} - (\alpha_{k+1} - 1)H_k - H_*\rangle\\ &\qquad + \frac{L\alpha_{k+1}}{2}\|Y_{k+1} - H_{k+1}\|_F^2.\end{aligned} \tag{48}$$

From (16), we get $\alpha_k^2 = \alpha_{k+1}^2 - \alpha_{k+1}$. By using this equality and multiplying both sides of (48) by α_{k+1}, we have

$$\begin{aligned}&\alpha_k^2\left(F(W^t, H_k) - F(W^t, H_*)\right)\\ &\quad - \alpha_{k+1}^2\left(F(W^t, H_{k+1}) - F(W^t, H_*)\right)\\ &\quad\geq \frac{L}{2}\Big(\|\alpha_{k+1}H_{k+1} - \alpha_{k+1}Y_{k+1}\|_F^2\\ &\qquad + 2\left\langle \alpha_{k+1}H_{k+1} - \alpha_{k+1}Y_{k+1}, \alpha_{k+1}Y_{k+1}\right.\\ &\qquad\left. - ((\alpha_{k+1} - 1)H_k + H_*)\right\rangle\Big).\end{aligned} \tag{49}$$

Since we have $\|B - A\|_F^2 + 2 < B - A, A - C >= \|B - C\|_F^2 - \|A - C\|_F^2$ for any matrices A, B and C, (49) can be rewritten as

$$\begin{aligned}&\alpha_k^2\left(F(W^t, H_k) - F(W^t, H_*)\right)\\ &\quad - \alpha_{k+1}^2\left(F(W^t, H_{k+1}) - F(W^t, H_*)\right)\\ &\quad\geq \frac{L}{2}\Big(\|\alpha_{k+1}H_{k+1} - ((\alpha_{k+1} - 1)H_k + H_*)\|_F^2\\ &\qquad - \|\alpha_{k+1}Y_{k+1} - ((\alpha_{k+1} - 1)H_k + H_*)\|_F^2\Big)\\ &\quad= \frac{L}{2}\Big(\|\alpha_{k+1}H_{k+1} - ((\alpha_{k+1} - 1)H_k + H_*)\|_F^2\\ &\qquad - \|\alpha_k H_k - ((\alpha_k - 1)H_{k-1} + H_*)\|_F^2\Big)\end{aligned} \tag{50}$$

where the last equality is according to (17). By varying the subscript in (50) from 0 to $k-1$ and summing up all these inequalities, we get

$$\begin{aligned}&F(W^t, H_0) - F(W^t, H_*) - \alpha_k^2\left(F(W^t, H_k) - F(W^t, H_*)\right)\\ &\geq \frac{L}{2}\left(\|\alpha_k H_k - ((\alpha_k - 1)H_k + H_*)\|_F^2 - \|H_0 - H_*\|_F^2\right)\\ &\geq -\frac{L}{2}\|H_0 - H_*\|_F^2.\end{aligned} \tag{51}$$

By substituting $H = H_*, Y = Y_0$ into (45), we have

$$\begin{aligned}
&F(W^t, H_0) - F(W^t, H_*) \\
&\quad \le L\langle Y_0 - H_0, Y_0 - H_*\rangle - \frac{L}{2}\|Y_0 - H_0\|_F^2 \\
&\quad = \frac{L}{2}\left(\langle Y_0 - H_*, 2Y_0 - 2H_0\rangle - \langle Y_0 - H_0, Y_0 - H_0\rangle\right) \\
&\quad = \frac{L}{2}\left(\langle Y_0 - H_*, Y_0 - H_* + H_* - H_0 + Y_0 - H_0\rangle \right.\\
&\qquad \left. -\langle Y_0 - H_0, Y_0 - H_0\rangle\right) \\
&\quad = \frac{L}{2}\left(\langle Y_0 - H_*, Y_0 - H_*\rangle + \langle Y_0 - H_*, H_* - H_0\rangle \right.\\
&\qquad \left. +\langle Y_0 - H_*, Y_0 - H_0\rangle - \langle Y_0 - H_0, Y_0 - H_0\rangle\right) \\
&\quad = \frac{1}{2}\left(\|Y_0 - H_*\|_F^2 + \langle Y_0 - H_*, H_* - H_0\rangle \right.\\
&\qquad \left. +\langle H_0 - H_*, Y_0 - H_0\rangle\right) \\
&\quad = \frac{1}{2}\left(\|Y_0 - H_*\|_F^2 + \langle H_0 - H_*, H_* - H_0\rangle\right) \\
&\quad = \frac{L}{2}\left(\|Y_0 - H_*\|_F^2 - \|H_0 - H_*\|_F^2\right). \qquad (52)
\end{aligned}$$

By combining (51) and (52), we have

$$\begin{aligned}
&\alpha_k^2\left(F(W^t, H_k) - F(W^t, H_*)\right) \\
&\quad \le F(W^t, H_0) - F(W^t, H_*) + \frac{L}{2}\|H_0 - H_*\|_F^2 \\
&\quad \le \frac{L}{2}\left(\|Y_0 - H_*\|_F^2 - \|H_0 - H_*\|_F^2\right) + \frac{L}{2}\|H_0 - H_*\|_F^2 \\
&\quad \le \frac{L}{2}\|Y_0 - H_*\|_F^2. \qquad (53)
\end{aligned}$$

By substituting $Y_0 = H^t$ into (53) and using

$$\alpha_k^2 \ge (k+2)/2$$

in [22], we get

$$\begin{aligned}
F(W^t, H_k) - F(W^t, H_*) &\le \frac{L\|H^t - H_*\|_F^2}{2\alpha_k^2} \\
&\le \frac{2L\|H^t - H_*\|_F^2}{(k+2)^2}.
\end{aligned}$$

This completes the proof. ■

Acknowledgment

The authors greatly thank the handling associate editor Professor Dr. Konstantinos Slavakis and anonymous reviewers for their constructive comments to this paper.

References

[1] A. Beck and M. Teboulle, “A fast iterative shrinkage-thresholding algorithm for linear inverse problems,” *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, 2009.

[2] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, “Algorithms and applications for approximate nonnegative matrix factorization,” *Computat. Statist. Data Anal.*, vol. 52, no. 1, pp. 155–173, 2007.

[3] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Belmont, MA: Athena Scientific, 1999.

[4] S. Bonettini, “Inexact block coordinate descent methods with application to the nonnegative matrix factorization,” *IMA J. Numer. Anal.*, 2011, 10.1093/imanum/drq024.

[5] D. Cai, X. He, X. Wu, and J. Han, “Nonnegative matrix factorization on manifold,” in *Proc. IEEE Int. Conf. Data Mining*, 2008, pp. 63–72.

[6] A. Cichocki, R. Zdunek, and S. Amari, “New algorithms for nonnegative matrix factorization in applications to blind source separation,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2006, vol. 5, pp. 621–624.

[7] C. Cieri, D. Graff, M. Liberman, N. Martey, and S. Strassel, “The TDT-2 text and speech corpus,” in *Proc. DARPA Broadcast News Workshop*, 1999.

[8] D. Tao, X. Li, X. Wu, and S. J. Maybank, “Geometric mean for subspace selection,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 2, pp. 260–274, 2009.

[9] L. Grippo and M. Sciandrone, “On the convergence of the block nonlinear Gauss-Seidel method under convex constraints,” *Operat. Res. Lett.*, vol. 26, p. 127C136.

[10] N. Guan, D. Tao, Z. Luo, and B. Yuan, “Manifold regularized discriminative nonnegative matrix factorization with fast gradient descent,” *IEEE Trans. Image Process.*, vol. 20, no. 7, pp. 2030–2048, Jul. 2011.

[11] N. Guan, D. Tao, Z. Luo, and B. Yuan, “Nonnegative patch alignment framework,” *IEEE Trans. Neural Netw.*, vol. 22, no. 8, pp. 1218–1230, Aug. 2011.

[12] L. X. Han, N. Michael, and P. Upendra, “Alternating projected Barzilai-Borwein methods for nonnegative matrix factorization,” *Electron. Trans. Numer. Anal.*, vol. 36, pp. 54–82, 2009.

[13] P. O. Hoyer, “Nonnegative sparse coding,” in *Proc. IEEE Workshop on Neural Netw. Signal Process.*, 2002, pp. 557–565.

[14] S. W. Ji and J. P. Ye, “An accelerated gradient method for trace norm minimization,” in *Proc. Int. Conf. Mach. Learn.*, 2009, vol. 382, pp. 457–464.

[15] J. Kim and H. Park, “Toward faster nonnegative matrix factorization: A new algorithm and comparisons,” in *Proc. 8th IEEE Int. Conf. Data Mining*, 2008.

[16] H. Kim and H. Park, “Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method,” *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 713–730, May 2008.

[17] D. Kim, S. Sra, and I. Dhillon, “Fast Newton-type methods for the least squares non-negative matrix approximation problem,” in *Proc. IEEE Int. Conf. Data Mining*, 2007, pp. 343–354.

[18] D. D. Lee and H. S. Seung, “Algorithms for nonnegative matrix factorization,” *Adv. Neural Inf. Process. Syst.*, vol. 12, pp. 556–562, 2000.

[19] D. D. Lewis, Y. M. Yang, T. G. Rose, and F. Li, “RCV1: A new benchmark collection for text categorization research,” *J. Mach. Learn. Res.* , vol. 5, pp. 361–397, 2004.

[20] C. J. Lin, “Projected gradient methods for non-negative matrix factorization,” *Neural Comput.*, vol. 19, pp. 2756–2779, 2007.

[21] C. J. Lin, “On the convergence of multiplicative update algorithms for nonnegative matrix factorization,” *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1589–1596, Nov. 2007.

[22] Y. Nesterov, “A Method of Solving A Convex Programming Problem with Convergence Rate $O\left(\frac{1}{k^2}\right)$,” *Soviet Math. Doklady*, vol. 27, no. 2, 1983.

[23] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Boston, MA: Kluwer Academic, 2004.

[24] Y. Nesterov, “Smooth minimization of non-smooth functions,” *Math. Programm.*, vol. 103, no. 1, pp. 127–152, 2005.

[25] P. Paatero and U. Tapper, “Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values,” *Environmetr.*, vol. 5, no. 2, pp. 111–126, 1994.

[26] V. P. Pauca, F. Shahnaz, M. W. Berry, and R. J. Plemmons, “Text mining using non-negative matrix factorization,” in *Proc. IEEE Int. Conf. Data Mining*, 2004, pp. 452–456.

[27] R. T. Rockafellar, *Convex Analysis*. Princeton, NJ: Princeton Univ. Press, 1970.

[28] M. Song, D. Tao, C. Chen, X. Li, and C. Chen, “Color to gray: Visual cue preservation,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 9, pp. 1537–1552, 2010.

[29] X. Tian, D. Tao, and Y. Rui, “Sparse transfer learning for interactive video search reranking,” *ACM Trans. Multimed. Comput., Commun., Appl.*, 2011.

[30] X. Wang, Z. Li, and D. Tao, “Subspaces indexing model on Grassmann manifold for image search,” *IEEE Trans. Image Process.*, vol. 20, no. 9, pp. 2627–2635, 2011.

[31] W. Xu, X. Liu, and Y. Gong, "Document clustering based on nonnegative matrix factorization," *ACM Special Interest Group on Inf. Retr.*, pp. 267–273, 2003.

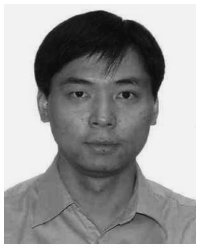
[32] J. Yu, D. Liu, D. Tao, and H. S. Seah, "Complex object correspondence construction in two-dimensional animation," *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3257–3269, 2011.

[33] R. Zdunek and A. Cichocki, "Non-negative matrix factorization with quasi-Newton optimization," in *Proc. 8th Int. Conf. Artif. Intell. Soft Comput.*, 2006, vol. 4029, pp. 870–879.

[34] T. Zhou and D. Tao, "Fast gradient clustering," in *Proc. NIPS 2009 Workshop on Discrete Optimiz. Mach. Learn.: Submodularity, Sparsity Polyhedra*, 2009, pp. 1–6.

Naiyang Guan received the B.S. and M.S. degrees from the National University of Defense Technology, China.

He is currently pursuing the Ph.D. degree in the School of Computer Science, National University of Defense Technology. From October 2009 to October 2010, he was a Visiting Student with the School of Computer Engineering, Nanyang Technological University, Singapore. He is currently a Visiting Scholar with the Centre for Quantum Computation and Information Systems and the Faculty of Engineering and Information Technology, University of Technology, Sydney, Australia. His current research interests include computer vision, image processing, and convex optimization.



Dacheng Tao (M'07–SM'12) is a Professor of Computer Science with the Centre for Quantum Computation and Information Systems and the Faculty of Engineering and Information Technology, University of Technology, Sydney. His interests include statistics and mathematics for data analysis problems in data mining, computer vision, machine learning, multimedia, and video surveillance. He has authored and coauthored more than 100 scientific articles.

Prof. Tao received the Best Theory/Algorithm paper runner-up award at IEEE ICDM'07.

Zhigang Luo received the B.S., M.S., and Ph.D. degrees from the National University of Defense Technology in 1981, 1993, and 2000, China, respectively.

He is currently a Professor with the School of Computer Science, National University of Defense Technology. His research interests include parallel computing, computer simulation, and bioinformatics.

Bo Yuan received the Bachelor degree from Peking University Medical School in 1983, China; the M.S. degree in biochemistry, and the Ph.D. degree in molecular genetics from the University of Louisville, KY, in 1990 and 1995, respectively.

He is currently a Professor with the Department of Computer Science and Engineering, Shanghai Jiao Tong University (SJTU). Before joining SJTU in 2006, he was a tenure-track Assistant Professor with the Ohio State University (OSU), while serving as a co-Director for the OSU Program in Pharmacogenomics. At OSU, he was the founding director for the OSU's Genome Initiative during the early 2000's, leading one of the only three independent efforts in the world (besides the Human Genome Project and the Celera company), having assembled and deciphered the entire human and mouse genomes. At SJTU, his research interests focus on biological networks, network evolution, stochastic process, biologically inspired computing, and bioinformatics, particularly on how these frameworks might impact the development of intelligent algorithms and systems.