A Fast Algorithm for Nonnegative Matrix Factorization and Its Convergence

Li-Xin Li, Lin Wu, Hui-Sheng Zhang, and Fang-Xiang Wu, *Senior Member, IEEE*

***Abstract*— Nonnegative matrix factorization (NMF) has recently become a very popular unsupervised learning method because of its representational properties of factors and simple multiplicative update algorithms for solving the NMF. However, for the common NMF approach of minimizing the Euclidean distance between approximate and true values, the convergence of multiplicative update algorithms has not been well resolved. This paper first discusses the convergence of existing multiplicative update algorithms. We then propose a new multiplicative update algorithm for minimizing the Euclidean distance between approximate and true values. Based on the optimization principle and the auxiliary function method, we prove that our new algorithm not only converges to a stationary point, but also does faster than existing ones. To verify our theoretical results, the experiments on three data sets have been conducted by comparing our proposed algorithm with other existing methods.**

***Index Terms*— Auxiliary function, convergence, multiplicative updates, nonnegative matrix factorization (NMF), optimization, stationary point.**

I. Introduction

UNSUPERVISED learning methods, such as principal component analysis (PCA) [1], [2], independent component analysis (ICA) [3], [4], and network component analysis (NCA) [5], [6] can be understood as factorizing a data matrix subject to different constraints [5], [6]. The differences mainly come from the constraints imposed. The PCA requires principal components being mutually orthogonal [2], the ICA requires components being mutually independent [4], whereas the NCA requires one of submatrices reflecting network structures [5]. As a result, the factors from different methods are shown to have very different representational properties. Lee and Seung [7] have shown that nonnegativity is a useful constraint for data matrix factorization that can learn the part representation of the whole data.

Manuscript received December 11, 2012; revised October 14, 2013; accepted December 21, 2013. Date of publication January 10, 2014; date of current version September 15, 2014. This work was supported in part by the Natural Science Foundation of Shaanxi Province, China, under Grant 2012JQ8025, in part by the Northwestern Polytechnical University Basic Research Fund under Grant FFR-NPU-JC201118 through LXL, and in part by the Natural Sciences and Engineering Research Council of Canada through FXW.

L.-X. Li and H.-S. Zhang are with the School of Electronics and Information, Northwestern Polytechical University, Xi'an 710072, China (e-mail: lilixin@nwpu.edu.cn; zhanghuisheng@nwpu.edu.cn).

L. Wu is with the Division of Biomedical Engineering, University of Saskatchewan, Saskatoon, SK S7N5A9, Canada (e-mail: liw557@mail.usask.ca).

F.-X. Wu is with the Department of Mechanical Engineering and Division of Biomedical Engineering, University of Saskatchewan, Saskatoon, SK S7N5A9, Canada (e-mail: faw341@mail.usask.ca).

Digital Object Identifier 10.1109/TNNLS.2013.2296627

Nonnegative matrix factorization (NMF) tries to find nonnegative factors with reduced ranks to approximate a given nonnegative data matrix [7]. Mathematically, given a nonnegative data matrix A with the size of $n \times m$, and a positive integer $r < \min(n, m)$, the NMF finds two nonnegative matrices $U = (u_{ik}) \in R^{n\times r}$ and $V = (v_{lj}) \in R^{r\times m}$, such that

$$A \approx UV.$$

There are several ways to measure the best approximation of A by UV, one of which has widely been used is to minimize the Euclidean distance between A and UV as follows:

$$\min_{U,V} f(U,V) = \min_{U,V} \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij} - (UV)_{ij})^2$$
$$\text{subject to} \quad u_{ik} \geq 0, v_{lj} \geq 0 \quad \forall i,k,l,j. \tag{1}$$

Note that each nonnegative constraint is subject to only a single variable. Although there are some other ways to measure the best approximation of A by UV such as the generalized Kullback–Leibler divergence between A and UV [7], this paper focuses on algorithms for solving the NMF formulated by optimization problem (1).

In this paper, we will use the following standard notation for the norm of a matrix and a vector. For a matrix $X = (x_{ij}) \in R^{n\times m}$, its Frobenius norm is defined as $\|X\|_F = \sqrt{\sum_{i=1}^{n}\sum_{j=1}^{m} x_{ij}^2}$. For a vector $x = (x_1, \ldots, x_n) \in R^n$, its 2-norm is defined as $\|x\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$. With this notation, the objective function in (1) can also be expressed in terms of the matrix Frobenius norm as

$$f(U,V) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(A_{ij} - (UV)_{ij})^2 = \frac{1}{2}\|A - UV\|_F^2$$

and furthermore

$$f(U,V) = \frac{1}{2}\|A - UV\|_F^2 = \frac{1}{2}\sum_{j=1}^{m}\|a_j - Uv_j\|_2^2 = \frac{1}{2}\sum_{i=1}^{n}\|a_i' - u_i V\|_2^2 \tag{2}$$

where a_j and v_j are the jth column vectors of matrix A and V, for $j = 1,\ldots, m$, respectively, whereas a_i' and u_i are the ith row vectors of matrix A and U for $i = 1, 2, \ldots, n$, respectively.

Paatero and Tapper [8] in 1994 first proposed the NMF (originally named positive matrix factorization) and further

developed the NMF in [9] and [10]. However, the NMF became a very popular unsupervised learning method because of [7] in 1999. Since then, the NMF has been widely applied to many areas such as biological data analysis [11]–[17], text mining [7], [18]–[20], image processing [7], [20], [21], and blind source separation [22], [23], and so on. Some pieces of software have also been developed for facilitating the applications of the NMF [24], [25]. A comprehensive review on the NMF algorithm and its applications can be found in [26] and references therein.

One of the main reasons that the NMF become popular after the reintroduction by Lee and Seung is that Lee and Seung propose a very simple multiplicative update algorithm for the NMF in [7] and [27]. Lee and Seung [27] have proved that their algorithm can guarantee the value of objective function in (1) is nonincreasing with increasing the number of iterations and finally converges to a stationary point. However, later on several researchers have pointed out that the Lee and Seung's algorithm in [27] does not converge to a stationary point in many cases [20], [28], [29] even if it may converge to a limit point. Note that although the constraints in (1) are convex, the objective function $f\ (U,\ V)$ in (1) is not convex in both U and V. Therefore, the optimization problem (1) is actually not convex in both U and V. Thus, the limit points of Lee and Seung's algorithm are not necessarily stationary points. However, the optimization problem (1) is convex in either U (or V) when V (or U) is given. Lin [29] proposes a modified multiplicative update algorithm for the NMF, which can guarantee to converge to a stationary point.

Another disadvantage of Lee and Seung's algorithm in [27] is that it is notoriously slow to converge when it converges. The Lin's modified algorithm in [29] is even slower to converge than Lee and Seung's algorithm in [27] as the former includes two small positive numbers and needs the extra calculations for the modified variables. Besides Lee and Seung's multiplicative update algorithm, Berry *et al.* [20] also mention other two types of algorithms for the NMF: 1) gradient decent algorithms and 2) alternating least squares algorithms. The gradient decent algorithms do not guarantee to converge or converge very slow without a careful choice of the search step size [20]. Actually, both Lee and Seung's algorithm and Lin's modified algorithm are the gradient decent algorithms with smart choices of search step sizes. The alternating least squares algorithms [8]–[10], [30] can be very fast with a good implementation, but the convergence is not warranted [20].

In summary, on the one hand, the NMF is a very important unsupervised machine learning method and widely used in many areas. On the other hand, the existing algorithms have their shortcomings. In this paper, we propose a new multiplicative update algorithm for the NMF and prove that our presented algorithm does not only converge to a stationary point, but also converges faster than existing multiplicative update algorithms [27], [29]. In Section II, we give the Karush–Kuhn–Tucher (KKT) optimality conditions of optimization problem (1), which the stationary point should be satisfied with. Then, we discuss several existing algorithms for solving optimization problem (1), especially both Lee and Seung's algorithm and Lin's modified algorithm. In Section III, we present our fast multiplicative update algorithm for solving optimization problem (1) and analyze its properties and convergence by using auxiliary functions [27], [29]. We also compare our proposed algorithm with the algorithms reviewed in Section II. To verify our theoretical results, in Section IV we apply our proposed algorithm, as well as both Lee and Seung's algorithm and Lin's modified algorithm to three data sets. The experimental results are analyzed and discussed. In Section V, we conclude this paper and discuss some related issues.

II. KKT Optimality Conditions and Existing Algorithms for NMF

Although the optimization problem (1) is not convex in both U and V, a local optimal point should be a stationary point according to the optimization principle [31]. By the definition, $(U,\ V)$ is a stationary point of the optimization problem (1) if it satisfies with KKT optimality conditions [31], which are as follows
for $\forall i, k, l, j$:

$$u_{ik} \geq 0, \qquad v_{lj} \geq 0 \tag{3a}$$

$$\frac{\partial f(U,V)}{\partial u_{ik}} \geq 0, \qquad \frac{\partial f(U,V)}{\partial v_{lj}} \geq 0 \tag{3b}$$

$$u_{ik}\frac{\partial f(U,V)}{\partial u_{ik}} = 0, \qquad v_{lj}\frac{\partial f(U,V)}{\partial v_{lj}} = 0 \tag{3c}$$

where

$$\frac{\partial f(U,V)}{\partial v_{lj}} = \sum_{i=1}^{n}\left(\sum_{s=1}^{r} u_{is}v_{sj} - a_{ij}\right)u_{il} \tag{4a}$$

$$\frac{\partial f(U,V)}{\partial u_{ik}} = \sum_{j=1}^{m}\left(\sum_{s=1}^{r} u_{is}v_{sj} - a_{ij}\right)v_{kj} \tag{4b}$$

are partial derivatives of $f\ (U,\ V)$ with respect to v_{lj} and u_{ik}, respectively.

Descent methods are widely used to find a local minimum of an objective function [31]. Typically, descent methods need to define a search step size and a search direction such that the value of the objective function is decreasing (at least nonincreasing) with increasing the number of iterations. As it is the steepest descent for minimizing the objective function, the negative gradient is naturally chosen as the search direction. The resulting methods are called the gradient descent algorithms. Applying the gradient descent method to the optimization problem (1) yields to the gradient descent algorithm (Algorithm 1).

In principle, on the one hand, the small positive search step sizes $\varepsilon(v_{lj}^t)$ and $\varepsilon(u_{ik}^t)$ can warrant that the value of objective function is decreasing (at least nonincreasing) with increasing the number of iterations. However, it would be very slow to converge with small search step sizes. On the other hand, the larger the search step size, the faster the value of objective function is decreasing. However, the updated variables may not satisfy with the constraints. Therefore, although the negative

Algorithm 1 [20]: Gradient Descent Algorithm

For t=1,2,…

$$v_{lj}^{t+1} = v_{lj}^{t} - \varepsilon(v_{lj}^{t}) \frac{\partial f(U^t, V^t)}{\partial v_{lj}^{t}} \qquad \forall l,j \tag{5a}$$

$$u_{ik}^{t+1} = u_{ik}^{t} - \varepsilon(u_{ik}^{t}) \frac{\partial f(U^t, V^t)}{\partial u_{ik}^{t}} \qquad \forall i,k \tag{5b}$$

gradient descent is known as the steepest descent, the gradient descent algorithm could very slowly converges to a local minimum or even not converge to it without a smart choice of search step sizes, especially for the optimization problems with constraints such as in optimization problem (1). In [21] and [32], the initial step size is set as 1, then multiplying them by a predefined number <1 at each subsequent iteration. This is simple, yet not ideal because there is no warrant that all updated elements of matrices U and V are nonnegative. In practice, many gradient descent algorithms just simply set all negative updated elements to zeros [18], [21], [33], [34], [36]. However, there has been no convergence theory to support this approach.

Therefore, to solve the optimization problem (1) with the gradient descent algorithms, one should define the search step sizes such that all updated elements of matrices U and V are nonnegative and the algorithm finally converges to a stationary point. Lee and Seung [27] set the search step sizes as follows:

$$\varepsilon_{\mathrm{LS}}(v_{lj}^{t}) = \frac{v_{lj}^{t}}{[(U^t)^T U^t V^t]_{lj}} \quad \forall l, j \tag{6a}$$

$$\varepsilon_{\mathrm{LS}}(u_{ik}^{t}) = \frac{u_{ik}^{t}}{[U^t V^t (V^t)^T]_{ik}} \quad \forall i, k. \tag{6b}$$

As a result, they obtain the multiplicative update algorithm (Algorithm 2) for solving the optimization problem (1).

Obviously this algorithm can guarantee that all updated elements are nonnegative if all initial values are nonnegative. Lee and Seung [27] prove that this algorithm converges to a stationary point. Because of its simplicity, this algorithm has played an important role in popularizing the applications of the NMF. However, there are several concerns with this algorithm. First of all, the initialization of this algorithm is a concern. Lin [32] has proved that for $t \geq 1$, all updated values of v_{lj}^{t} and u_{ik}^{t} are positive for all l, j, i, k if A has neither zero columns nor zero rows and initial values of v_{lj} and u_{ik} for all *l, j, i, k* are positive. Obviously, positive matrices U and V might not be the optimal solution of (1), especially when matrix A has some zero elements. Furthermore, in majority of applications of the NMF, more zero elements in matrices U and V are expected. On the other hand, from (7a) and (7b), for $t \geq 1$, the updated values of v_{lj}^{t} and u_{ik}^{t} are always zeros for some *l, j, i, k* if their corresponding initial values are zeros. As a result, although the algorithm can converge to a limit point, yet it may not be a stationary point. For example, if $v_{lj} = 0$ yet $\partial f(U, V)/\partial v_{lj} < 0$, the condition (3b) is not satisfied. This has been numerically illustrated in [29]. In addition, this algorithm is not well defined if denominators in (7a) or (7b) are zero.

Algorithm 2 [27]: Multiplicative Update Algorithm

For t=1,2,…

$$v_{lj}^{t+1} = v_{lj}^{t} \frac{[(U^t)^T A]_{lj}}{[(U^t)^T U^t V^t]_{lj}} \qquad \forall l,j \tag{7a}$$

$$u_{ik}^{t+1} = u_{ik}^{t} \frac{[A(V^{t+1})^T]_{ik}}{[U^t V^{t+1} (V^{t+1})^T]_{ik}} \qquad \forall i,k \tag{7b}$$

Lin [29] choose the following more sophisticated search step sizes to improve Algorithm 2:

$$\varepsilon_{\mathrm{Lin}}(v_{lj}^{t}) = \frac{\tilde{v}_{lj}^{t}}{[(U^t)^T U^t \tilde{V}^t]_{lj} + \delta} \quad \forall l, j \tag{8a}$$

$$\varepsilon_{\mathrm{Lin}}(u_{ik}^{t}) = \frac{\tilde{u}_{ik}^{t}}{[\tilde{U}^t V^{t,n} (V^{t,n})^T]_{ik} + \delta} \quad \forall i, k \tag{8b}$$

where

$$\tilde{v}_{lj}^{t} \equiv \begin{cases} v_{lj}^{t}, & \text{if } \partial f(U^t, V^t)/\partial v_{lj}^{t} \geq 0 \\ \max(v_{lj}^{t}, \sigma), & \text{if } \partial f(U^t, V^t)/\partial v_{lj}^{t} < 0 \end{cases}$$

and

$$\tilde{u}_{ik}^{t} \equiv \begin{cases} u_{ik}^{t}, & \text{if } \partial f(U^t, V^t)/\partial u_{ik}^{t} \geq 0 \\ \max(u_{ik}^{t}, \sigma), & \text{if } \partial f(U^t, V^t)/\partial u_{ik}^{t} < 0 \end{cases}$$

and both σ and δ are predefined small positive numbers. As a result, Lin obtains the modified multiplicative update algorithm (Algorithm 3).

Lin [29] has proved that for $t \geq 1$, all updated values of v_{lj}^{t} and u_{ik}^{t} in Algorithm 3 are nonnegative for all *l, j, i, k* if all initial values of v_{lj} and u_{ik} for all *l, j, i, k* are nonnegative and it converges to a stationary point.

Computational complexity is another concern for both Algorithms 2 and 3. In both Algorithms 2 and 3, the main cost is dominated by computing both $U^{\mathrm{T}}UV$ and UVV^T or both $\partial f(U^t, V^t)/\partial v_{lj}^{t}$ and $\partial f(U^t, V^t)/\partial u_{ik}^{t}$, as $r < \min(n, m)$, each of which takes the order $O(\mathrm{mnr})$ of operations at each iteration. Therefore, the computational complexity order for these both algorithms is #iterations $\times$ $O(\mathrm{mnr})$. In practice, as Lin's [29] experiments have shown, Algorithm 3 is even slower to converge than Algorithm 2, which is already thought to be too slow [20]. Actually, comparing them one can obviously see that Algorithm 3 has to take more operations at each iteration than Algorithm 2. In addition, the search step size (8) for Algorithm 3 is not greater than the search step size (6) for Algorithm 2 in practice, which is another reason that Algorithm 3 is slower to converge than Algorithm 2.

III. Fast Multiplicative Update Algorithm and its Convergence Theory

Before presenting our new algorithm, we would like to discuss about the parameter r in the NMF, which is the number of columns of matrix U and the number of rows of matrix V. In most applications, one may know the range of values of r if not knowing the exact value. In some situations, one may want to find the minimal value of r such that the approximation of the NMF is good enough. Under no circumstances, one

Algorithm 3 [29]: Modified Multiplicative Update Algorithm

1) Given $\sigma > 0$ and $\delta > 0$. Initialize $v_{lj}^1 \geq 0$ and $u_{ik}^1 \geq 0$, for all *l, j, i, k.*
2) For t=1, 2, …
   a) If (U^t, V^t) is stationary, i.e., satisfying with (3), stop
   Else

$$v_{lj}^{t,n} = v_{lj}^t - \frac{\widetilde{v}_{lj}^t}{[(U^t)^T U^t \widetilde{V}^t]_{lj} + \delta} \frac{\partial f(U^t, V^t)}{\partial v_{lj}^t} \quad \forall l,j \qquad (9a)$$

$$u_{ik}^{t,n} = u_{ik}^t - \frac{\widetilde{u}_{ik}^t}{[\widetilde{U}^t V^{t,n} (V^{t,n})^T]_{ik} + \delta} \frac{\partial f(U^t, V^t)}{\partial u_{ik}^t} \quad \forall i,k \qquad (9b)$$

   b) Normalize $v_{lj}^{t,n}$ and $u_{ik}^{t,n}$ to v_{lj}^{t+1} and u_{ik}^{t+1} respectively so that the sum of each column of matrix U^{t+1} is one.

expects the resultant matrix U has some zero columns or the resultant matrix V has some zero rows. In practice, if these situations happen, one can reduce the value of r by removing zero columns in U and zero rows in V. However, during the iterations, some columns of U or rows of V might be temporally zeros and we should pay special attention to these cases.

A. Fast Multiplicative Update Algorithm

We present the following fast multiplicative update algorithm (Algorithm 4) for solving the optimization problem (1). When $u_l^t = 0$, v_{lj}^t can take any value while the value of objective function $f(U, V)$ will not change. We simply set $v_{lj}^t = 0$ for all j. Furthermore, we will prove such a setting also satisfies the KKT optimality conditions (3). For the same reason, when $v_k^t = 0$, we set $u_{ik}^t = 0$.

Similar to Algorithms 2 and 3, the computational complexity is dominated by calculating $\partial f(U^t, V^t)/\partial v_{lj}^t$ and $\partial f(U^t, V^t)/\partial u_{ik}^t$, each of which takes the order $O(\text{mnr})$ of operations. Therefore, the complexity order of our algorithm is #iterations $\times$ $O(\text{mnr})$, which is the same as those of Algorithms 2 and 3. However, the more detailed analyses can conclude that at each iteration our presented Algorithm 4 just has a few more operations than Algorithm 2 does, but much less operations than Algorithm 3 does.

One may note that in Algorithm 4 operations $v_{lj}^{t+1} = \max(0, \bar{v}_{lj}^{t+1})$ and $u_{ik}^{t+1} = \max(0, \bar{u}_{ik}^{t+1})$ project negative updates of $\bar{v}_{lj}^{t+1}$ and $\bar{u}_{ik}^{t+1}$ to zeros. It seems nothing new than existing methods in [18], [21], [33], and [34]. However, our update formula $\bar{v}_{lj}^{t+1}$ and $\bar{u}_{ik}^{t+1}$ are different from those in [18], [21], [33], and [34]. More importantly, in the following, we prove that Algorithm 4 does not only converge to a stationary point of the optimization problem (1), but also converges to it faster than Algorithms 2 and 3.

After comparing the convergence speeds of our presented Algorithm 4 with Algorithms 2 and 3 in terms of their search step sizes, we conclude the following.

Theorem 1: Our presented Algorithm 4 is (in most cases, strictly) faster than Algorithms 2 and 3.

Algorithm 4 Fast Multiplicative Update Algorithm for NMF

1) Initialize $u_{ik}^1 \geq 0$, $v_{lj}^1 \geq 0$ for all *i, k, l, j*
2) For *t*=1,2,….,
   If (U^t, V^t) is stationary, i.e., satisfying with (3), stop
   Else
   If $u_l^t = 0$, $v_{lj}^t = 0$, for all j, where u_l^t is the l-th column vector of matrix U^t
   Else

$$v_{lj}^{t+1} = \max(0, \bar{v}_{lj}^{t+1}) \qquad (10a')$$

   where

$$\bar{v}_{lj}^{t+1} = v_{lj}^t - \frac{1}{\left\|u_l^t\right\|^2} \frac{\partial f(U^t, V^t)}{\partial v_{lj}^t} \quad \forall l,j \qquad (10a)$$

   If $v_k^t = 0$, $u_{ik}^t = 0$, for all i, where v_k^t is the k-th row vector of matrix V^t
   Else

$$u_{ik}^{t+1} = \max(0, \bar{u}_{ik}^{t+1}) \qquad (10b')$$

   where

$$\bar{u}_{ik}^{t+1} = u_{ik}^t - \frac{1}{\left\|v_k^t\right\|^2} \frac{\partial f(U^t, V^t)}{\partial u_{ik}^t}, \forall i,k \qquad (10b)$$

Proof: In Algorithm 4, the search step sizes are

$$\varepsilon_{Wu}(v_{lj}^t) = \frac{1}{\left\|u_l^t\right\|^2} \quad \forall l, j \qquad (11a)$$

$$\varepsilon_{Wu}(u_{ik}^t) = \frac{1}{\left\|v_i^t\right\|^2} \quad \forall i,k. \qquad (11b)$$

Note that

$$\begin{aligned}\varepsilon_{\text{Lin}}(v_{lj}^t) &= \frac{\tilde{v}_{lj}^t}{[(U^t)^T U^t \tilde{V}^t]_{lj} + \delta} < \frac{\tilde{v}_{lj}^t}{\sum_{k=1}^{r} \left(\sum_{i=1}^{n} u_{il}^t u_{ik}^t \right) \tilde{v}_{kj}^t} \\ &= \frac{\tilde{v}_{lj}^t}{\sum_{\substack{k=1 \\ k \neq l}}^{r} \left(\sum_{i=1}^{n} u_{il}^t u_{ik}^t \right) \tilde{v}_{kj}^t + \sum_{i=1}^{n} (u_{il}^t)^2 \tilde{v}_{lj}^t} \\ &\leq \frac{1}{\left\|u_l^t\right\|^2} = \varepsilon_{Wu}(v_{lj}^t) \qquad (12a)\end{aligned}$$

and

$$\begin{aligned}\varepsilon_{\text{LS}}(v_{lj}^t) &= \frac{v_{lj}^t}{[(U^t)^T U^t V^t]_{lj}} = \frac{v_{lj}^t}{\sum_{k=1}^{r} \left(\sum_{i=1}^{n} u_{il}^t u_{ik}^t \right) v_{kj}^t} \\ &= \frac{v_{lj}^t}{\sum_{\substack{k=1 \\ k \neq l}}^{r} \left(\sum_{i=1}^{n} u_{il}^t u_{ik}^t \right) v_{kj}^t + \sum_{i=1}^{n} (u_{il}^t)^2 v_{lj}^t} \\ &\leq \frac{1}{\left\|u_l^t\right\|^2} = \varepsilon_{Wu}(v_{lj}^t). \qquad (12b)\end{aligned}$$

In most cases

$$\sum_{\substack{k=1\\k\neq l}}^{r}\left(\sum_{i=1}^{n}u_{il}^{t}u_{ik}^{t}\right)v_{kj}^{t}$$

and

$$\sum_{\substack{k=1\\k\neq l}}^{r}\left(\sum_{i=1}^{n}u_{il}^{t}u_{ik}^{t}\right)v_{kj}^{t}$$

are strict positive and thus the inequality is strict. Therefore, the proof of Theorem 1 is complete.

Furthermore, in Algorithms 2 and 3, if $v_{lj}^{t}\neq 0$ ($u_{ik}^{t}\neq 0$), the search step sizes are proportional to the values of v_{lj}^{t} (u_{ik}^{t}) from (6) and (8). Therefore, both Algorithms 2 and 3 update the small values of v_{lj} (u_{ik}) very slowly. For $v_{lj}^{t}=0$ ($u_{ik}^{t}=0$), the search step sizes in Algorithm 2 are zeros and thus elements $v_{lj}(u_{ik})$ have no way to be updated even if the KKT optimality conditions are not satisfied. For $v_{lj}^{t}=0(u_{ik}^{t}=0)$ and $\partial f(U^{t},V^{t})/\partial v_{lj}^{t}<0$ ($\partial f(U^{t},V^{t})/\partial u_{ik}^{t}<0$), although the search step sizes in Algorithm 3 are nonzeros, they are very small positive numbers because $\tilde{v}_{lj}^{t}$ ($\tilde{u}_{ik}^{t}$) $=\sigma$ are very small, $\sigma=10^{-8}$, as Lin [29] suggested. Therefore, even though Algorithm 3 can update the zero values of v_{lj} (u_{ik}), it does very slowly. From (11), the search step sizes of our Algorithm 4 are independent of the values of v_{lj}^{t} (u_{ik}^{t}). Therefore, for the small values of v_{lj}^{t} (u_{ik}^{t}), the values of v_{lj} (u_{ik}) can be largely updated, in particular for $v_{lj}^{t}=0(u_{ik}^{t}=0)$ and $\partial f(U^{t},V^{t})/\partial v_{lj}^{t}<0$ ($\partial f(U^{t},V^{t})/\partial u_{ik}^{t}<0$).

B. Proof of Convergence

Similar to [27] and [29], the auxiliary function is adopted to provide the proof of convergence. Thus, we first introduce the concept of auxiliary function and its important property below.

Definition 1 [27]: $G(v,v')$ is an auxiliary function for $F(v)$ if the following conditions:

$$G(v,v')\geq F(v)\quad\text{and}\quad G(v,v)=F(v)$$

are satisfied.

The concept of auxiliary function is very useful in proof of convergence because of the following property.

Lemma 1 [27]: If $G(v,v')$ is an auxiliary function for $F(v)$, then F is nonincreasing under the following update:

$$v^{t+1}=\arg\min_{v}G(v,v^{t}). \tag{13}$$

The proof is straightforward by considering

$$F(v^{t+1})\leq G(v^{t+1},v^{t})\leq G(v^{t},v^{t})=F(v^{t}).$$

Lee and Seung [27] and Lin [29] try to update a whole column of matrix U or a whole row of matrix V at the same time in Algorithms 2 and 3, respectively, taking the use of (2) of the objective function in the optimization problem (1). As a result, the auxiliary function is a function of vectors. Subsequently, they have to employ matrix operations to analyze the convergence of their algorithms, which makes the analysis unnecessarily complicated. In this paper, we analyze the convergence of Algorithm 4 by considering to update an individual element of matrix U or V at one time. It seems simple, yet Algorithm 4 is faster than Algorithms 2 and 3 as shown in Section III-A, which will also be numerically demonstrated in Section IV through three data sets.

Theorem 2: The objective function of problem (1) is nonincreasing and actually decreasing as fast as possible under the update rules of our presented Algorithm 4.

Proof: In the following, we derive the formulas for updating the $(l,\ j)$th element of V in our presented Algorithm 4. Given matrices U^{t} and V^{t}, if we only change the $(l,\ j)$th element of V^{t} to v_{lj} while keeping other elements in U^{t} and V^{t} unchanged. Denote the changed matrix V^{t} by V. Note that the objective function of optimization problem (1) is actually a quadratic function of each of all individual elements in matrices U and V. Therefore, for given matrices U^{t} and V^{t} $f(U^{t},\ V)$ is an exactly quadratic function of v_{lj} as follows:

$$f(U^{t},V)=f(U^{t},V^{t})+\frac{\partial f(U^{t},V^{t})}{\partial v_{lj}^{t}}(v_{lj}-v_{lj}^{t})+\frac{1}{2}\frac{\partial^{2}f(U^{t},V^{t})}{\partial(v_{lj}^{t})^{2}}(v_{lj}-v_{lj}^{t})^{2} \tag{14}$$

where

$$\frac{\partial f(U^{t},V^{t})}{\partial v_{lj}^{t}}=\sum_{i=1}^{n}\left(\sum_{s=1}^{r}u_{is}^{t}v_{sj}^{t}-a_{ij}\right)u_{il}^{t} \tag{15a}$$

$$\frac{\partial^{2}f(U^{t},V^{t})}{\partial(v_{lj}^{t})^{2}}=\sum_{i=1}^{n}(u_{il}^{t})^{2}=\left\|u_{l}^{t}\right\|^{2}. \tag{15b}$$

Here, we define the following function with respect to v_{lj}:

$$G(U^{t},V^{t},V)=f(U^{t},V^{t})+\frac{\partial f(U^{t},V^{t})}{\partial v_{lj}^{t}}(v_{lj}-v_{lj}^{t})+\frac{1}{2}(\left\|u_{l}^{t}\right\|^{2}+\alpha_{lj})(v_{lj}-v_{lj}^{t})^{2}. \tag{16}$$

By the definition of auxiliary function, we can see that for any $\alpha_{lj}\geq 0$, function (16) is the auxiliary function for function (14). Therefore, we obtain the update rule for v_{lj} as follows:

$$\bar{\bar{v}}_{lj}^{t+1}=\arg\min_{v_{lj}}G(U^{t},V^{t},V)=v_{lj}^{t}-\frac{1}{\left(\left\|u_{l}^{t}\right\|^{2}+\alpha_{lj}\right)}\frac{\partial f(U^{t},V^{t})}{\partial v_{lj}^{t}} \tag{17}$$

By Lemma 1, replacing v_{lj}^{t} by $\bar{\bar{v}}_{lj}^{t+1}$ calculated by (17) for any $\alpha_{lj}\geq 0$ and keeping other elements in U^{t} and V^{t} unchanged, the value of objective function f in the optimization problem (1) is nonincreasing. Note that when $\alpha_{lj}=0$, formula (17) becomes formula (10a), that is $\bar{\bar{v}}_{lj}^{t+1}=\bar{v}_{lj}^{t+1}$. The update rule (17) is actually a gradient descent rule with the search step size

$$\varepsilon(v_{lj}^{t})=\frac{1}{\left(\left\|u_{l}^{t}\right\|^{2}+\alpha_{lj}\right)}. \tag{18}$$

The larger the value of $\varepsilon(v_{lj}^t)$, the faster the value of the objective function in (1) is decreasing. For the given matrix U^t, the largest value of $\varepsilon(v_{lj}^t)$ is $1/\|u_l^t\|^2$ at $\alpha_{lj}=0$. However, when the value of $\varepsilon(v_{lj}^t)$ is large, the updated value of $\bar{\bar{v}}_{lj}^{t+1}$ by (17) could be negative, which violates the KKT optimality conditions in (3a). Therefore, we need to carefully choose the value of α_{lj} to make sure that all the KKT optimal conditions (3a) would be satisfied while the value of the objective function in (1) is decreasing as fast as possible.

Here, we consider two cases: 1) $\|u_l^t\|^2=0$ and 2) $\|u_l^t\|^2\neq 0$, yet the updated value of $\bar{\bar{v}}_{lj}^{t+1}$ is negative. For Case 1), if $\|u_l^t\|^2=0$, then we have $u_{il}^t=0$ for all $i=1,\ldots,n$. Furthermore, from (15a) and (15b) both $\partial f(U^t,V^t)/\partial v_{lj}^t$ and $\partial^2 f(U^t,V^t)/\partial^2 v_{lj}^t$ are zero, which indicates that the function $f(U^t,V)$ in (14) is independent of v_{lj}. Therefore, by taking any updated value of v_{lj}^{t+1}, the value of function $f(U^t,V)$ will not change. As $\partial f(U^t,V^t)/\partial v_{lj}^t=0$, the KKT optimality conditions (3b) and (3c) for v_{lj} are satisfied. To satisfy the KKT optimality conditions (3a) for v_{lj}, we simply take $v_{lj}^{t+1}=0$.

For Case 2), we need to carefully choose the nonnegative α_{lj} such that the updated value of $\bar{\bar{v}}_{lj}^{t+1}$ calculated by (17) is nonnegative and the value of the objective function in (1) is decreasing as fast as possible. We need to consider two cases again: a) $\partial f(U^t,V^t)/\partial v_{lj}^t<0$ and b) $\partial f(U^t,V^t)/\partial v_{lj}^t\geq 0$. For Case a), the updated value of $\bar{\bar{v}}_{lj}^{t+1}$ calculated by (17) is positive for any value of α_{lj}. Thus, we take $\alpha_{lj}=0$ to have the largest search step size $1/\|u_l^t\|^2$ and then formula (17) becomes formula (10a). Note that when $v_{lj}^t=0$, in this case $(\partial f(U^t,V^t)/\partial v_{lj}^t<0)$, the (l,j)-th element of V still has a chance to be updated and the updated value of v_{lj}^{t+1} is positive. But in Algorithm 2, as long as $v_{lj}^t=0$ for some t, then for $t+1,t+2,\ldots$, the (l,j)-th element of V has no chance anymore for being updated even if $\partial f(U^t,V^t)/\partial v_{lj}^t<0$ is true, which violates the KKT optimality conditions (3b).

For Case b), the inequality

$$\frac{1}{\|u_l^t\|^2}\frac{\partial f(U^t,V^t)}{\partial v_{lj}^t}\geq 0 \tag{19}$$

is true. If $v_{lj}^t=0$, we can choose $\alpha_{lj}=+\infty$ in (17) to get $v_{lj}^{t+1}=0$. We can also get $v_{lj}^{t+1}=0$ by formula

$$v_{lj}^{t+1}=\max(0,\bar{v}_{lj}^{t+1})$$

where $\bar{v}_{lj}^{t+1}$ is calculated by (10a) and is nonpositive. If $v_{lj}^t>0$, we should choose α_{lj} to make sure that the inequality

$$v_{lj}^t-\frac{1}{\left(\|u_l^t\|^2+\alpha_{lj}\right)}\frac{\partial f(U^t,V^t)}{\partial v_{lj}^t}\geq 0 \tag{20}$$

is true. Solving (20) and considering $\alpha_{lj}\geq 0$ yields

$$\alpha_{lj}\geq\max\left(0,\frac{1}{v_{lj}^t}\frac{\partial f(U^t,V^t)}{\partial v_{lj}^t}-\|u_l^t\|^2\right).$$

To have the largest possible search step size, we take

$$\alpha_{lj}=\max\left(0,\frac{1}{v_{lj}^t}\frac{\partial f(U^t,V^t)}{\partial v_{lj}^t}-\|u_l^t\|^2\right).$$

If $\alpha_{lj}=0$, the value of $\bar{\bar{v}}_{lj}^{t+1}$ calculated by (17) [i.e., (10a)] is nonnegative.

If $\alpha_{lj}=1/v_{lj}^t\partial f(U^t,V^t)/\partial v_{lj}^t-\|u_l^t\|^2>0$, then the value of $\bar{\bar{v}}_{lj}^{t+1}$ calculated by (17) is exactly 0, whereas the value of $\bar{v}_{lj}^{t+1}$ calculated by (10a) is nonpositive. Therefore, again we can have

$$v_{lj}^{t+1}=\max(0,\bar{v}_{lj}^{t+1}).$$

In summary, for Case b), we have the formulas (10a) and (10b′) to update v_{lj}, which guarantee that the updated value of v_{lj}^{t+1} is nonnegative and the value of the objective function is nonincreasing and actually is decreasing as fast as possible.

By the similar process, the formulas for updating the (i,k)-th element of U in our presented Algorithm 4 can be derived. Therefore, the proof of Theorem 2 is complete.

In the following, we prove that the sequence of matrices $\{(U^t,V^t),t=1,2,\ldots.\}$ produced by Algorithm 4 has at least one limit point, which is also a stationary point.

Theorem 3: The sequence of matrices $\{(U^t,V^t),t=1,2,\ldots.\}$ produced by Algorithm 4 has at least one limit point.

Proof: By Theorem 2, the sequence $\{f(U^t,V^t),t=1,2,\ldots.\}$ is nonincreasing. As function $f(U,V)$ is bounded below, $\lim_{t\to\infty}f(U^t,V^t)$ exists and is finite. As matrix A is bounded, $\|U^tV^t\|_F^2$ is bounded for all $t\geq 1$, which indicates that $\sum_{s=1}^r u_{is}^t v_{sj}^t$ is bounded, for all $t\geq 1$ and all i,j.

Assume that for some (l,j), $\lim_{t\to\infty}v_{lj}^t=\infty$. If there exist i and t_1 such that $u_{il}^t\neq 0$ for all $t\geq t_1$, then $\lim_{t\to\infty}\sum_{s=1}^r u_{is}^t v_{sj}^t=\infty$, which contradicts against the boundedness of $\sum_{s=1}^r u_{is}^t v_{sj}^t$. If for all i, $u_{il}^t=0$, from Algorithm 4, $v_{lj}^t=0$, which contradicts against $\lim_{t\to\infty}v_{lj}^t=\infty$. Thus, we can conclude that for all (l,j), v_{lj}^t is bounded. Similar arguments can conclude that for all (i,k), u_{ik}^t is bounded. Therefore, the sequence $\{(U^t,V^t),t=1,2,\ldots.\}$ is bounded in matrix Frobenius norm. Thus, $\{(U^t,V^t),t=1,2,\ldots.\}$ has at least one limit point according to Bolzano–Weierstrass theorem [37].

Theorem 4: Assume that $\{(U^t,V^t),t=1,2,\ldots.\}$ is produced by Algorithm 4 and converges to (U^*,V^*). Then, (U^*,V^*) satisfies with the KKT optimality conditions (3).

Proof: From Algorithm 4, for all (l,j), if the lth column vector of matrix U^* is zero, that is, $u_l^*=0$, then $v_{lj}^*=0$ and $\partial f(U^*,V^*)/\partial v_{lj}^*=0$ for all j. Thus, the KKT optimality conditions (3) for v_{lj}^* are satisfied. If $u_l^*\neq 0$, we have

$$v_{lj}^*=\max\left(0,v_{lj}^*-\frac{1}{\|u_l^*\|^2}\frac{\partial f(U^*,V^*)}{\partial v_{lj}^*}\right) \tag{21}$$

if $v_{lj}^*=0$, from (21), $\partial f(U^*,V^*)/\partial v_{lj}^*\geq 0$ is true, and thus the KKT optimality conditions (3) for v_{lj}^* are satisfied.

If $v_{lj}^*>0$, from (21), we have

$$v_{lj}^*=v_{lj}^*-\frac{1}{\|u_l^*\|^2}\frac{\partial f(U^*,V^*)}{\partial v_{lj}^*}$$

which means that $\partial f(U^*,V^*)/\partial v_{lj}^*=0$, and thus the KKT optimality conditions (3) for v_{lj}^* are satisfied again. Similarly, we can prove that for all (i,k), the KKT optimality conditions (3) for u_{ik}^* are satisfied.

IV. EXPERIMENTS

Although our proposed algorithm has been theoretically proven to coverge faster than existing algorithms, this section presents numerical experiments on three data sets to further illustrate its performance in practice. Three algorithms are programmed in MATLAB R2013a and run on a computer with the following specifications: a processor of Intel Core 2 Quad CPU Q9450 at 2.66 GHZ 2.67 GHZ and a RAM of 4 GB (3.72 GB usable). The code for Algorithm 3 is adopted from [29] with $\sigma = \delta = 10^{-8}$, whereas the codes for Algorithms 2 and 4 are created by ourselves. In our experiments, all three programs are run with two synthetic and one real-life image data sets. The two synthetic data sets were created as $a_{ij} = |N(0, 1)|$, where $N(0, 1)$ stands the standard normal distribution. One mediate-size synthetic data set has $(n, r, m) = (40, 10, 250)$, whereas another large-size synthetic data set has $(n, r, m) = (400, 50, 2500)$. The real-life image data set is the ORL face image database and is preprocessed in the same way as they have been used to evaluate the methods in [21], [27] and [29].

In our experiments, initial values of U and V are set in the two following cases.

Case I: set $u_{is} = |N(0, 1)|$ and $v_{sj} = |N(0, 1)|$.

Case II: set $u_{is} = |N(0, 1)|$ and $v_{sj} = |N(0, 1)|$, and randomly set 20% of elements in U as zeros.

In this paper, initial values in Case II are considered in order to verify that Algorithm 2 does not converge to the best solution in this case as stated in Section II and [29].

To fairly compare them, all three algorithms for the same data set start with the same sets of 30 different initial values with Case I or II. The algorithms are judged to converge if the following criterion is met:

$$\frac{\left|\text{OBJ}^k - \text{OBJ}^{k-1}\right|}{\text{OBJ}^k} \leq \varepsilon \tag{22}$$

where OBJ^k is the value of objective function (2) at iteration step k, and ε is a small positive number and is set as 10^{-6} in this paper.

Tables 1–3 report the average results for each of three data sets over 30 different initial values of U and V with Cases I and II. They include the average CPU time, the average number of iterations, the average values, and the standard deviation of the objective function when algorithms converge. From Tables, one can see that for all three data sets, the average values of objective function with Case II is much larger than those with Case I for Algorithm 2, which is consistent with our conclusions that Algorithm 2 may not converge to the stationary point when the initial values of matrix U or V are zeros as they always stay as zeros. On the other hand, for all three data sets, the average values of objective function with Case II is smaller than or comparable to those with Case I for Algorithms 3 and 4, which means that Algorithms 3 and 4 can find the optimal solution even if the initial values of matrix U or V are zeros.

Most importantly, from Tables 1–3, first it is clear that for all three data sets and two cases of initial values our proposed Algorithm 4 converges faster than Algorithms 2 and 3 measured by CPU times and the number of iterations as theoretically proved in Section III. Second, the average values of objective function from our proposed Algorithm 4 are smaller than those from Algorithms 2 and 3, which indicates that our proposed Algorithm 4 can find a better solution than Algorithms 2 and 3 when they converge.

Third, the standard deviation of the values of objective function from our proposed Algorithm 4 is smaller than those from Algorithms 2 and 3. This indicates that our proposed Algorithm 4 is also less sensitive to the initial values of U and V than Algorithms 2 and 3.

TABLE I
RESULTS OF THE MEDIATE-SIZE SYNTHETIC DATA SET

	Algorithm2	Algorithm3	Algorithm4
initial values in case I			
CPUTime(s)	0.70	1.05	**0.25**
Iteration	1519.83	1814.57	**303.06**
OBJ.ave	1114.33	1113.00	**1110.05**
OBJ.std	2.29	2.31	**0.71**
initial values in case II			
CPUTime(s)	0. 71	1.35	**0.32**
Iteration	1595.67	1338.90	**399.26**
OBJ.ave	1165.19	1150.37	**1147.83**
OBJ.std	3.85	1.90	**1.07**

TABLE II
RESULTS OF THE LARGE-SIZE SYNTHETIC DATA SET

	Algorithm2	Algorithm3	Algorithm4
initial values in case I			
CPUTime(s)	110.14	129.80	**106.60**
Iteration	2730.37	2978.53	**776.80**
OBJ.ave	149450.1	149354.3	**148681.0**
OBJ.std	35.39	43.57	**27.79**
initial values in case II			
CPUTime(s)	91.66	130.12	**88.18**
Iteration	2518.97	3330.40	**741.97**
OBJ.ave	149914.2	149290.5	**148639.6**
OBJ.std	34.86	48.45	**22.33**

TABLE III
RESULTS OF THE REAL-LIFE IMAGE DATA SET

	Algorithm2	Algorithm3	Algorithm4
initial values in case I			
CPUTime(s)	716.89	860.59	**280.97**
Iteration	5751.8	6529.97	**1288.70**
OBJ.ave	13.22	13.19	**13.08**
OBJ.std	0.034	0.033	**0.016**
initial values in case II			
CPUTime(s)	706.72	1053.66	**322.02**
Iteration	5173.73	7295.63	**1032.73**
OBJ.ave	16.74	13.20	**13.08**
OBJ.std	0.048	0.028	**0.016**

V. Conclusion

In this paper, we have studied the NMF formulated in terms of the optimization problem (1). After analyzing the existing algorithms, we have proposed a new multiplicative update algorithm for solving this problem. Based on optimization principle and auxiliary function method, we have proved that our algorithm not only converges to a stationary point, but also does faster than existing algorithms. The experimental results on three data sets have also verified our theoretical results. The main contributions of this paper are threefolds. First, we have proposed a new multiplicative algorithm for solving the NMF and have theoretically and numerically proved that the proposed algorithm converges faster than existing algorithms. Second, we have theoretically proved that any limit point of the proposed algorithm is a stationary point. Third, the numerical experiments have also shown that the proposed algorithm is more robust to the initial values. The methodologies adopted in this paper could be applied to study other constraint optimization problems. For example, as one of directions of future work, we could study the convergence of algorithms for solving the NMF formulated in terms of divergence measurement in [7]. Another direction of future work is that we can apply our proposed algorithm to solve problems in biological data analysis, image process, text mining, and so on.

Acknowledgment

The authors would like to thank the editors and reviewers for valuable suggestions and comments for improving the original manuscript.

References

[1] H. Hotelling, "Analysis of a complex of statistical variables into principal components," *J. Educ. Psychol.*, vol. 24, no. 6, pp. 417–441, Sep. 1933.

[2] M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer-Verlag, 2007.

[3] A. Hyvarinen, "Fast and robust fixed-point algorithms for independent component analysis," *IEEE Trans. Neural Netw.*, vol. 10, no. 3, pp. 626–634, May 1999.

[4] A. Hyvärinen, J. Karhunen, and E. Oja, *Independent Component Analysis*. New York, NY, USA: Wiley, 2001

[5] J. C. Liao, R. Boscolo, Y.-L. Yang, L. M. Tran, C. Sabatti, and V. P. Roychowdhury, "Network component analysis: Reconstruction of regulatory signals in biological systems," in *Proc. Nat. Acad. Sci. United States Amer.*, vol. 100. Dec. 2003, pp. 15522–15527.

[6] R. Boscolo, C. Sabatti, J. C. Liao, and V. P. Roychowdhury, "A generalized framework for network component analysis," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 2, no. 4, pp. 289–301, Oct./Dec. 2005.

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

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

[9] P. Paatero, "Least squares formulation of robust non-negative factor analysis," *Chemometrics Intell. Lab. Syst.*, vol. 37, no. 1, pp. 23–35, May 1997.

[10] P. Paatero, "The multilinear engine—A table-driven least squares program for solving multilinear problems, including the *n*-way parallel factor analysis model," *J. Comput. Graph. Statist.*, vol. 8, no. 4, pp. 1–35, 1999.

[11] L. P. Tian, L. Liu, and F.-X. Wu, "Matrix decomposition methods in bioinformatics," *Current Bioinf.*, vol. 8, no. 2, pp. 259–266, Apr. 2013.

[12] H. Kim and H. Park, "Sparse non-negative matrix factorizations via alternating non-negativity-constrained least squares for microarray data analysis," *Bioinformatics*, vol. 23, no. 12, pp. 1495–1502, 2007.

[13] K. Devarajan, "Nonnegative matrix factorization: An analytical and interpretive tool in computational biology," *PLoS Comput. Biol.*, vol. 4, no. 7, p. e1000029, Jul. 2008.

[14] D. Greene, G. Cagney, N. Krogan, and P. Cunningham, "Ensemble non-negative matrix factorization methods for clustering protein–protein interactions," *Bioinformatics*, vol. 24, no. 15, pp. 1722–1728, 2008.

[15] Y. Li and A. Ngom, "Nonnegative least-squares methods for the classification of high dimensional biological data," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 10, no. 2, pp. 447–456, Mar./Apr. 2013.

[16] Y. Li and A. Ngom, "Classification approach based on non-negative least squares," *Neurocomputing*, vol. 118, pp. 41–57, Oct. 2013.

[17] Q. Qi, Y. Zhao, M. Li, and R. Simon, "Non-negative matrix factorization of gene expression profiles: A plug-in for BRB-ArrayTools," *Bioinformatics*, vol. 25, no. 4, pp. 545–547, 2009.

[18] F. Shahnaz, M. W. Berry, V. Pauca, and R. J. Plemmons, "Document clustering using nonnegative matrix factorization," *Inf. Process. Manag.*, vol. 42, no. 2, pp. 373–86, Mar. 2006.

[19] W. Xu, X. Liu, and Y. Gong, "Document clustering based on nonnegative matrix factorization," in *Proc. 26th Annu. Int. ACM SIGIR Conf. Res. Develop. Inf. Retr.*, 2003, pp. 267–73.

[20] M. Berry, M. Browne, A. Langville, V. Pauca, and R. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Comput. Statist. Data Anal.*, vol. 52, no. 1, pp. 155–173, 2007.

[21] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, Jan. 1, 2014.

[22] A. Cichocki, S. Amari, R. Zdunek, R. Kompass, G. Hori, and Z. He, "Extended SMART algorithms for non-negative matrix factorization," in *Artificial Intelligence and Soft Computing* (Lecture Notes in Computer Science), vol. 4029. New York, NY, USA: Springer-Verlag, 2006, pp. 548–562.

[23] G. Zhou, Z. Yang, S. Xie, and J. M. Yang, "Online blind source separation using incremental nonnegative matrix factorization with volume constraint," *Trans. Neural Netw.*, vol. 22, no. 4, pp. 550–560, Apr. 2011.

[24] R. Gaujoux and C. Seoighe, "A flexible R package for nonnegative matrix factorization," *BMC Bioinf.*, vol. 11, p. 367, Jul. 2010.

[25] Y. Li and A. Ngom, "The non-negative matrix factorization toolbox for biological data mining," *BMC Sour. Code Biol. Med.*, vol. 8, no. 1, pp. 1–10, 2013.

[26] Y. X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 6, pp. 1336–1353, Jun. 2013.

[27] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *NIPS*. Cambridge, MA, USA: MIT Press, 2000.

[28] E. Gonzalez and Y. Zhang, "Accelerating the Lee-Seung algorithm for nonnegative matrix factorization," Dept. Comput. Appl. Math., Rice Univ., Houston, TX, USA, Tech. Rep. TR-05-02, 2005.

[29] C.-J. Lin, "On the convergence of multiplicative update algorithms for nonnegative matrix factorization," *IEEE Trans. Neural Netw.*, vol. 18, no. 6, pp. 1589–1597, Nov. 2007.

[30] 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.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2009.

[32] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, no. 10, pp. 2756–2779, Oct. 2007.

[33] M. Chu, F. Diele, R. Plemmons, and S. Ragni. (2004). Optimality, computation, and interpretations of nonnegative matrix factorizations [Online]. Available: http://www.wfu.edu/~plemmons

[34] P. Pauca, J. Piper, and R. Plemmons, "Nonnegative matrix factorization for spectral data analysis," *Linear Algebra Appl.*, vol. 416, no. 1, pp. 29–47, Jul. 2006.

[35] C. Ding, X. He, and H. Simon, "On the equivalence of nonnegative matrix factorization and spectral clustering," in *Proc. 5th SIAM Int. Conf. Data Mining*, Newport Beach, CA, USA, 2005, pp. 606–615.

[36] Z. Yang and E. Oja, "Linear and nonlinear projective nonnegative matrix factorization," *IEEE Trans. Neural Netw.*, vol. 21, no. 5, pp. 734–749, May 2010.

[37] P. M. Fitzpatrick, *Advanced Calculus*, 2nd ed. Providence, RI, USA: AMS, 2006.

Li-Xin Li received the B.Sc. and M.Sc. degrees in communication engineering, and the Ph.D. degree in control theory and its applications from Northwestern Polytechnical University (NPU), Xi'an, China, in 2001, 2004, and 2008, respectively.

He was a Post-Doctoral Fellow with NPU from 2008 to 2010. He is currently an Assistant Professor with the School of Electronics and Information, NPU. He has published more than 30 technical papers in refereed journals and conference proceedings. His current research interests include image analysis, wireless communication, information theory, and anti-jamming technology.

Lin Wu received the B.Sc. degree in computer science and technology from Central South University, Hunan, China, in 2012. He is currently pursuing the Ph.D. degree with the Division of Biomedical Engineering, University of Saskatchewan, Saskatoon, SK, Canada.

His current research interests include the analysis of large-scale biological data, modeling, analysis, and control of molecular biological networks.

Hui-Sheng Zhang received the B.Sc. and M.Sc. degrees in electronic engineering from Northwestern Polytechnical University (NPU), Xi'an, China, in 1977 and 1986, respectively.

He is currently a Full Professor and Ph.D. Advisor with the School of Electronics and Information, NPU. He has published more than 80 technical papers in refereed journals and conference proceedings. His current research interests include mobile communication, anti-jamming technology, and image analysis.

Prof. Zhang is a member of the Teaching Steering Committee of Ministry of Education of the China. He received the Distinguished Teacher Award of Shaanxi Province in 2006.

Fang-Xiang Wu (M'06–SM'11) received the B.Sc. and M.Sc. degrees in applied mathematics from the Dalian University of Technology, Dalian, China, in 1990 and 1993, respectively, the Ph.D. degree in control theory and its applications from Northwestern Polytechnical University, Xi'an, China, in 1998, and the Ph.D. degree in biomedical engineering from the University of Saskatchewan (U of S), Saskatoon, SK, Canada, in 2004.

He was a Post-Doctoral Fellow with the Laval University Medical Research Center, Quebec, QC, Canada, from 2004 to 2005. He is currently a Full Professor of bioengineering with the Department of Mechanical Engineering and the Graduate Chair of the Division of Biomedical Engineering at the U of S. He has published over 180 technical papers in refereed journals and conference proceedings. His current research interests include computational and systems biology, genomic and proteomic data analysis, biological system identification and parameter estimation, and applications of control theory to biological systems.

Dr. Wu is serving as the Editorial Board Member of five international journals and Guest Editor of several international journals, and Program Committee Chair or member of several international conferences. He has reviewed papers for many international journals.