International Journal of Engineering, Mathematical and Physical Sciences
ISSN: 2517-9934
Vol:8, No:1, 2014

# Convergence Analysis of an Alternative Gradient Algorithm for Non-Negative Matrix Factorization

Chenxue Yang, Mao Ye, Zijian Liu, Tao Li, Jiao Bao

*Abstract*—Non-negative matrix factorization (NMF) is a useful computational method to find basis information of multivariate nonnegative data. A popular approach to solve the NMF problem is the multiplicative update (MU) algorithm. But, it has some defects. So the columnwisely alternating gradient (cAG) algorithm was proposed. In this paper, we analyze convergence of the cAG algorithm and show advantages over the MU algorithm. The stability of the equilibrium point is used to prove the convergence of the cAG algorithm. A classic model is used to obtain the equilibrium point and the invariant sets are constructed to guarantee the integrity of the stability. Finally, the convergence conditions of the cAG algorithm are obtained, which help reducing the evaluation time and is confirmed in the experiments. By using the same method, the MU algorithm has zero divisor and is convergent at zero has been verified. In addition, the convergence conditions of the MU algorithm at zero are similar to that of the cAG algorithm at non-zero. However, it is meaningless to discuss the convergence at zero, which is not always the result that we want for NMF. Thus, we theoretically illustrate the advantages of the cAG algorithm.

*Keywords*—Non-negative matrix factorizations, convergence, cAG algorithm, equilibrium point, stability.

## I. Introduction

NON-NEGATIVE matrix factorization (NMF) is a useful computational method to find basis information of multivariate nonnegative data [1, 6]. And it has such linear and non-negative approximate representation. Given a non-negative data matrix $V \in R^{m\times n}$, NMF finds an approximate factorization

$$V \approx WH, \tag{1}$$

where $W$ and $H$ are $m \times r$ and $r \times n$ non-negative matrices, respectively. Usually, the positive integer $r$ is chosen as smaller than $m$ or $n$, so that $W$ and $H$ are smaller than the original matrix $V$.

In 1994, NMF was first proposed in [4], when it was called positive matrix factorization. Recently, NMF has been paid much attention and applied to many application areas including image databases [5], text data mining [7, 8], subsystem identification [9], spectral data analysis [10], speech processing [11, 12], blind source separation (BSS) [14], etc.

Chenxue Yang, Mao Ye,Tao Li and Jiao Bao are with school of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, P.R. China and State Key Lab. for Novel Software Technology, Nanjing University, P.R. China. Email addresses: yang.chenxue@163.com(Chenxue Yang),yem_mei29@hotmail.com(Mao Ye)
Zijian Liu is with school of science, chongqingjiaotong university, chongqing, 400074, P.R. China and Department of Mathematics, Hangzhou Normal University, Hangzhou, 310036, PR China. Email address: hbliuzijian@126.com

To find the matrices $W$ and $H$, the problem in (1) is commonly reformulated as minimizing the following cost function:

$$f(W,H) := \frac{1}{2} \parallel V - WH \parallel_F^2, \tag{2}$$

where $\parallel \bullet \parallel_F$ represents the Frobenius norm.

There have been various types of NMF techniques in the literatures, such as gradient descent methods [15], alternating least squares [3], the popular multiplicative update algorithm (MU) [1] and so on. But, the algorithm in [1], proposed by Lee and Seung, fails to approach to a stationary point. This defect is showed in [2] by numerical examples. After that, for the defect that the MU algorithm appears inner iteration and zero divisor, Lin [13] proposed three algorithms on NMF problem, which are the columnwisely alternating gradient (cAG) algorithm, the columnwisely alternating projected gradient (cAPG) algorithm and the elementwisely alternating projected gradient (eAPG) algorithm. And the latter two algorithms are based on the cAG algorithm. Since there does not exist exact mathematical proof of convergence of the above algorithms. Therefore, in this paper, we analyze the convergence of the cAG algorithm and show the advantages over the MU algorithm theoretically.

Currently, there are many classical methods on the convergence analysis of the algorithms for the non-negative matrix factorization, such as projected gradient approaches [17], bound optimization [18] and so on. What's more, some algorithms to accelerate the convergence are designed in [19]. In this paper, we will utilize the stability of the equilibrium point to prove the convergence of the cAG algorithm. First, we prove the existence of the equilibrium point by Squeeze Theorem. Then, a classic model, which was proposed in [20], is used to obtain the equilibrium point. At last, using the method in [16], we construct the invariant sets to constrain the area of the initializations to guarantee the integrity of the convergence analysis.

Using the theory of the previous paragraph, we have that the cAG algorithm has not zero equilibrium point, which verifies the algorithm has not zero divisor. Meanwhile, we obtain that the cAG algorithm is locally convergent, and realize that good initialization can improve the speed and accuracy of the algorithm. If the initializations are not in the invariant sets, the $W$ and $H$ are approaching to be negative in the iterations. Similarly, the MU algorithm has zero divisor and is convergent at zero has been verified. In addition, the convergence conditions of the MU algorithm at zero are similar to that of the cAG algorithm at non-zero. However, it is meaningless to discuss the convergence at

zero, which is not always the result that we want. Thus, we theoretically illustrate the advantages of the cAG algorithm.

Our contributions can be summarized as follows. First, we prove the convergence of the cAG algorithm and obtain the structure of the equilibrium points, which lay a foundation for better understanding of the cAG algorithm. Second, we show the advantages of the cAG algorithm theoretically. Finally, we realize that the conditions of the initialization constraints can accelerate the convergence of the algorithm.

This paper is structured as follows. A brief overview of the MU algorithm and the cAG algorithm are given in Section II. In Section III, we prove the existence of the equilibrium points of the cAG algorithm and obtain their expressions by using the classic model in [20]. In Section IV, we construct the invariant sets to constrain the area of initializations. We prove the convergence of the cAG algorithm in Section V. In Section VI, the simulations of the cAG algorithm confirm our theories and the advantages over the MU algorithm are showed theoretically. Finally, a conclusion is given in Section VII.

## II. Non-negative Matrix Factorization Algorithms

The multiplicative update (MU) algorithm is proposed by Lee and Seung [1] to solve the NMF problem (2), which is a variation of the gradient descent method using the following rules:

$$H_{kj} \longleftarrow H_{kj}\frac{[W^TV]_{kj}}{[W^TWH]_{kj}} \tag{3}$$

for $1 \le k \le r$ and $1 \le j \le n$;

$$W_{ik} \longleftarrow W_{ik}\frac{[VH^T]_{ik}}{[WHH^T]_{ik}} \tag{4}$$

for $1 \le i \le m$ and $1 \le k \le r$.

The MU algorithm is not defined well if one of the elements of $W^TWH$ or $WHH^T$ is zero. In addition, it may appear inner iteration. To avoid these defects, Lin [13] proposes the cAG algorithm which is described generally as the following.

When $W$ is fixed, H is updated column by column. For the $j$-th column, define

$$\begin{aligned}\eta_j^* &= \arg\min_{\eta_j}\frac{1}{2}\|V_j - W(H_j - \eta_j[\text{grad}(H)]_j)\|^2 \\ &= \frac{\|\text{grad}(H)]_j\|^2}{\|W\text{grad}(H)]_j\|^2},\end{aligned} \tag{5}$$

where

$$\text{grad}(H) := \frac{\partial f(W,H)}{\partial H} = W^T(WH - V), \tag{6}$$

$V_j$, $H_j$ and $[\text{grad}(H)]_j$ denote the $j$-th column of the matrices $V$, $H$, and $\text{grad}(H)$, respectively. For the $i$-th element of $H_j$, the algorithm for $H$ is

$$h_{ij} \longleftarrow h_{ij} - \eta_{ij}[\text{grad}(H)]_{ij}, \tag{7}$$

where $1 \le i \le m$, $1 \le j \le n$,

$$\eta_{ij} = \min_{[\text{grad}(H)]_{ij}>0}\Big\{\eta_j^*, \frac{h_{ij}}{[\text{grad}(H)]_{ij}}\Big\}, \tag{8}$$

and $[\text{grad}(H)]_{ij}$ denotes the $(i,j)$th element of the gradient matrix $\text{grad}(H)$.

Similarly, when $H$ is fixed, define

$$\begin{aligned}\xi_i^* &= \arg\min_{\xi_i}\frac{1}{2}\|V_i^T - H^T(W_i^T - \xi_i[\text{grad}(W)]_i^T)\|^2 \\ &= \frac{\|[\text{grad}(W)]_i^T\|^2}{\|H^T[\text{grad}(W)]_i^T\|^2},\end{aligned} \tag{9}$$

where

$$\text{grad}(W) := \frac{\partial f(W,H)}{\partial W} = (WH - V)H^T, \tag{10}$$

$V_i^T$, $W_i^T$ and $[\text{grad}(W)]_i^T$ stand for the $i$th column of $V^T$, $W^T$, and $[\text{grad}(W)]^T$, respectively. For the $j$-th element of $W_i^T$, the algorithm for $W$ is

$$w_{ij} \longleftarrow w_{ij} - \xi_{ij}[\text{grad}(W)]_{ji}^T, \tag{11}$$

where $1 \le i \le m$, $1 \le j \le n$,

$$\xi_{ij} = \min_{[\text{grad}(W)]_{ij}>0}\Big\{\xi_i^*, \frac{w_{ij}}{[\text{grad}(W)]_{ij}}\Big\}, \tag{12}$$

and $[\text{grad}(W)]_{ij}$ denotes the $(i,j)$th element of the gradient matrix $[\text{grad}(W)]$.

The choices of $\eta_j$ and $\xi_i$ ensure the nonnegativity of $H$ and $W$. From (5) and (9), the denominators are not zero, unless $W[\text{grad}(H)]_j = 0$ or $H^T[\text{grad}(W)_i^T] = 0$ for some $j$ or $i$.

## III. Existence of Equilibrium Points

Similarly, as the method in [16], before analyzing the convergence of the cAG algorithm, we need to prove the existence of the equilibrium points and obtain the structures of them. Both $H_j$ and $W_i$ will be updated alternately using (7) and (11), but they are computed separately. Thus, it is reasonable to consider $W$ being fixed firstly to prove that the equilibrium point of $H$ exits, and obtain its structure by using the model in [20]. Then, we fix $H$ to prove that the equilibrium point of $W$ exits, and obtain its structure in the same way.

For simplicity, we denote the following notation for any $\{x_{ij}\}_{rn}$, which means that $x_{ij}$ stands for the (i, j) element of a matrix $X \in R^{r\times n}$, and $1 \le i \le r$, $1 \le j \le n$.

By calculating the matrix, we define

$$W^TV = \Big\{\sum_{p=1}^{m} w_{ip}v_{pj}\Big\}_{rn} \equiv \{a_{ij}\}_{rn},$$

$$W^TW = \Big\{\sum_{p=1}^{m} w_{ip}w_{pi}\Big\}_{rr} \equiv \{b_{ik}\}_{rr}$$

and

$$[\text{grad}(H)]_j = \begin{pmatrix} \sum_{k=1}^{r} b_{1k}h_{kj} - a_{1j} \\ \sum_{k=1}^{r} b_{2k}h_{kj} - a_{2j} \\ \cdots \\ \sum_{k=1}^{r} b_{rk}h_{kj} - a_{rj} \end{pmatrix} = \begin{pmatrix} A_1 \\ A_2 \\ \cdots \\ A_r \end{pmatrix}.$$

From (5), it follows that

$$\eta_j^* = \frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2}. \quad (13)$$

If $\eta_j^* > \frac{h_{ij}}{[\text{grad}(H)]_{ij}}$, then $\eta_{ij} = \frac{h_{ij}}{[\text{grad}(H)]_{ij}}$. From (7), for the $i$-th element of $H_j$, there is

$$h_{ij} \longleftarrow h_{ij} - \frac{h_{ij}}{[\text{grad}(H)]_{ij}}[\text{grad}(H)]_{ij}\ . \quad (14)$$

Then, we have

$$h_{ij} \longleftarrow h_{ij} - h_{ij}\frac{(\sum_{k=1}^{r} b_{ik}h_{kj} - a_{ij})}{\sum_{k=1}^{r} b_{ik}h_{kj} - a_{ij}} = 0. \quad (15)$$

If $\eta_j^* \leq \frac{h_{ij}}{[\text{grad}(H)]_{ij}}$, then $\eta_{ij} = \eta_j^* = \frac{\|[\text{grad}(H)]_j\|^2}{\|W[\text{grad}(H)]_j\|^2}$. Similarly, the learning rule will be

$$h_{ij} \longleftarrow h_{ij} - \frac{\|[\text{grad}(H)]_j\|^2}{\|W[\text{grad}(H)]_j\|^2}[\text{grad}(H)]_j. \quad (16)$$

Then, we have the following algorithm

$$h_{ij} \longleftarrow h_{ij} - A_i\frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2}. \quad (17)$$

From (8) and the above analysis, to discuss the convergence of algorithm (7) which is the cAG algorithm for H, we only need to discuss the convergence of algorithm (17).

**Definition 1.** *For the algorithm (17), a point $h_{ij} \in R$ is called an equilibrium if and only if it satisfies*

$$A_i\frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} = 0. \quad (18)$$

**Theorem 1.** *Equation (18) has not zero solution i.e., $h_{ij} \neq 0$.*

**Proof.** The contradiction method is used to show this theorem.

From (18), the denominator is not zero, i.e.,

$$\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2 > 0. \quad (19)$$

Thus, there exits a $p^\star$, such that

$$[\sum_{i=1}^{r}(w_{p^\star i}A_i)]^2 > 0. \quad (20)$$

Since $\eta_j^* \leq \frac{h_{ij}}{[\text{grad}(H)]_{ij}}$, we obtain

$$\frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} \leq \frac{h_{ij}}{\sum_{k=1}^{r} b_{ik}h_{kj} - a_{ij}}.$$

Let's assume that (18) has zero solution, i.e., $h_{ij} = 0$. From the above equation, it follows that

$$0 \leq \frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} = 0.$$

Then, we have $A_i = 0$, $i = 1, 2, ...r$. Obviously,

$$[\sum_{i=1}^{r}(w_{p^\star i}A_i)]^2 = 0, \quad (21)$$

which is contradictory with (20). Therefore, (18) has not zero solution. □

From Theorem 1, it verifies that the cAG algorithm has not zero divisor, which is one of the advantages over the MU algorithm.

Denote

$$f(h_{ij}) = A_i\frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2}, \quad (22)$$

where

$$A_i = \sum_{k=1}^{r} b_{ik}h_{kj} - a_{ij} = \sum_{q=1}^{i-1} b_{iq}h_{qj} + \sum_{q=i+1}^{r} b_{iq}h_{qj} + b_{ii}h_{ij} - a_{ij}. \quad (23)$$

Let's assume $H_j$ except $h_{ij}$, $a_{ij}$ and $b_{ij}$ in $W^TV$ and $W^TW$ are constants. In addition, in each iteration, both $w_{ij}$ and $h_{ij}$ will be updated alternately. Thus, it is reasonable to consider $w_{ij}$ as a constant here for the convergent analysis of $h_{ij}$. Similarly, for the analysis of $w_{ij}$, we consider $h_{ij}$ as a constant.

**Theorem 2.** *There exists $h_{ij}^{(0)} \in [0, \infty)$, such that $f(h_{ij}^{(0)}) = 0$, i.e., $h_{ij}^{(0)}$ is the equilibrium of algorithm (17).*

**Proof.** Obviously, $f(h_{ij})$ is a continuous function. From (20), there exits a $i^\star$, such that $w_{p^\star i^\star}A_{i^\star} \neq 0$. Thus, we have

$$\sum_{i=1}^{r}(A_i)^2 > 0.$$

From (19), it follows that

$$\frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} > 0. \quad (24)$$

For the initialization of A in algorithm (17), let $h_{ij}^{(1)} > 0$ and $\sum_{k=1}^{r} b_{ik}h_{kj}^{(1)} < a_{ij}$. From (23), we obtain $A_i^{(1)} < 0$. Thus, from (22), there is

$$f(h_{ij}^{(1)}) = A_i^{(1)}\frac{\sum_{i=1}^{r}(A_i^{(1)})^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i^{(1)})]^2} < 0.$$

Increasing $h_{ij}^{(1)}$, we can obtain another point $h_{ij}^{(2)}$, such that $\sum_{k=1}^{r} b_{ik}h_{kj}^{(2)} > a_{ij}$. Similarly,

$$f(h_{ij}^{(2)}) = A_i^{(2)} \frac{\sum_{i=1}^{r}(A_i^{(2)})^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i^{(2)})]^2} > 0.$$

According to the median theorem, there exists $h_{ij}^{(0)}$, $h_{ij}^{(1)} < h_{ij}^{(0)} < h_{ij}^{(2)}$, such that

$$f(h_{ij}^{(0)}) = A_i^{(0)} \frac{\sum_{i=1}^{r}(A_i^{(0)})^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i^{(0)})]^2} = 0.$$

□

From Theorem 2, algorithm (17) exists the equilibrium point $h_{ij}^{(0)}$. Next, we need to obtain the structure of $h_{ij}^{(0)}$ to analyze the stability of algorithm (17). From the classic model in [20], we have the following lemma.

**Lemma 1.** *In difference system $x(t+1) - x(t) = F(x(t))$, there exists an equilibrium point $x^\star$, if $F_1(x) \le F(x) \le F_2(x)$, where $F_1(x)$ and $F_2(x)$ are monotonous, which satisfies $\eta_1 F_2(x^\star) - F(x^\star) = F(x^\star) - \eta_2 F_1(x^\star) = 0$.*

The Lemma 1 is used to obtain the structure of the equilibrium point. For simplicity, we denote

$$a_1 = \min_{1\le i\le r, 1\le j\le n}\{a_{ij}\},\ a_2 = \max_{1\le i\le r, 1\le j\le n}\{a_{ij}\}, \quad (25)$$

$$b_1 = \min_{1\le i,k\le r}\{b_{ik}\},\ b_2 = \max_{1\le i,k\le r}\{b_{ik}\}, \quad (26)$$

$$w_1 = \min_{1\le p\le m, 1\le i\le r}\{w_{pi}\},\ w_2 = \max_{1\le p\le m, 1\le i\le r}\{w_{pi}\}, \quad (27)$$

$$\sum_{k=1}^{r} h_{kj} = \sum_{q=1}^{i-1} h_{qj} + \sum_{q=i+1}^{r} h_{qj} + h_{ij} = P + h_{ij}. \quad (28)$$

We have the following theorem.

**Theorem 3.** *There exists a non-zero equilibrium point of algorithm (17), for some constants $0 \le \eta_1 \le 1$ and $\eta_2 \ge 1$, the structure of the equilibrium point is $h_{ij} = \frac{a_1}{\eta_1 b_2} - P$ or $h_{ij} = \frac{a_2}{\eta_2 b_1} - P$ in the condition $\eta_2 a_1 b_1 = \eta_1 a_2 b_2$.*

**Proof.** For the term $A_i = \sum_{k=1}^{r} b_{ik}h_{kj} - a_{ij}$, from (25) and (26), it follows that

$$A_i \le b_2(\sum_{k=1}^{r} h_{kj}) - a_1.$$

From (28), there is

$$b_1(P + h_{ij}) - a_2 \le A_i \le b_2(P + h_{ij}) - a_1. \quad (29)$$

Using Lemma 1 to obtain the solutions of (18), both sides of the inequality (29) must satisfy the following conditions, respectively,

$$b_2(P + h_{ij}) - a_1 \ge 0 \quad (30)$$

and

$$b_1(P + h_{ij}) - a_2 \le 0. \quad (31)$$

From the above inequalities, there exists two cases,

$$(b_1 + b_2)(P + h_{ij}) \ge (a_1 + a_2) \quad (32)$$

and

$$(a_1 + a_2) > (b_1 + b_2)(P + h_{ij}). \quad (33)$$

For case 1, from the inequality (29), it follows that

$$[b_1(P + h_{ij}) - a_2]^2 \le (A_i)^2 \le [b_2(P + h_{ij}) - a_1]^2.$$

From (27), we obtain

$$w_1^2 r^2 [b_1(P + h_{ij}) - a_2]^2 \le [\sum_{i=1}^{r}(w_{pi}A_i)]^2 \le w_2^2 r^2 [b_2(P + h_{ij}) - a_1]^2.$$

In the above inequalities, only $h_{ij}$ is variable and all the others are constants, there is

$$\begin{aligned} &\frac{[b_1(P + h_{ij}) - a_2]^3}{mrw_2^2[b_2(P + h_{ij}) - a_1]^2} \\ &\le A_i \frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} \\ &\le \frac{[b_2(P + h_{ij}) - a_1]^3}{mrw_1^2[b_1(P + h_{ij}) - a_2]^2}. \end{aligned} \quad (34)$$

For the given initializations of W and H, we can choose two constants $0 \le \eta_1 \le 1$ and $\eta_2 \ge 1$, such that

$$\begin{aligned} &\frac{[\eta_2 b_1(P + h_{ij}) - a_2]^3}{mrw_2^2[b_2(P + h_{ij}) - a_1]^2} \\ &\le A_i \frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} \\ &\le \frac{[\eta_1 b_2(P + h_{ij}) - a_1]^3}{mrw_1^2[b_1(P + h_{ij}) - a_2]^2}. \end{aligned} \quad (35)$$

Base on Lemma 1, we analyze the following two equations,

$$\frac{[\eta_1 b_2(P + h_{ij}) - a_1]^3}{mrw_1^2[b_1(P + h_{ij}) - a_2]^2} = 0 \quad (36)$$

and

$$\frac{[\eta_2 b_1(P + h_{ij}) - a_2]^3}{mrw_2^2[b_2(P + h_{ij}) - a_1]^2} = 0. \quad (37)$$

Equation (36) has solution

$$h_{ij} = \frac{a_1}{\eta_1 b_2} - P. \quad (38)$$

And equation (37) has solution

$$h_{ij} = \frac{a_2}{\eta_2 b_1} - P. \quad (39)$$

For case 2, there is

$$\begin{aligned} &\frac{[b_2(P + h_{ij}) - a_1]^2}{mrw_2^2[b_1(P + h_{ij}) - a_2]} \\ &\le A_i \frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} \\ &\le \frac{[b_1(P + h_{ij}) - a_2]^2}{mrw_1^2[b_2(P + h_{ij}) - a_1]}. \end{aligned} \quad (40)$$

Choosing two constants $0 \leq \eta_1 \leq 1$ and $\eta_2 \geq 1$ which are different from $\eta_1$ and $\eta_2$ in the inequality (35), it follows that

$$\begin{aligned} &\frac{[\eta_1 b_2(P+h_{ij})-a_1]^2}{mrw_2^2[b_1(P+h_{ij})-a_2]} \\ &\leq A_i \frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2} \\ &\leq \frac{[\eta_2 b_1(P+h_{ij})-a_2]^2}{mrw_1^2[b_2(P+h_{ij})-a_1]}. \end{aligned} \tag{41}$$

Similarly, we get the following two solutions,

$$h_{ij} = \frac{a_1}{\eta_1 b_2} - P \tag{42}$$

and

$$h_{ij} = \frac{a_2}{\eta_2 b_1} - P. \tag{43}$$

Although $\eta_1$ and $\eta_2$ in (42) and (43) are different from $\eta_1$ and $\eta_2$ in (38) and (39), but the structures of the solutions in (18) are the same.

From Theorem 2 and the above analysis, there exists an equilibrium point for the algorithm (17) if the non-zero solutions of (18) satisfy

$$\frac{a_1}{\eta_1 b_2} - P = \frac{a_2}{\eta_2 b_1} - P. \tag{44}$$

The above equation can be simplified to

$$\eta_2 a_1 b_1 = \eta_1 a_2 b_2. \tag{45}$$

Therefore, for the algorithm (17), the structure of the equilibrium point is (37) or (38) which satisfies (45). □

Analogously, for a fixed H, we can get similar results with respect to W. Similarly, we denote the following notation,

$$VH^T = \Big\{\sum_{j=1}^{n} v_{pj}h_{ji}\Big\}_{mr} \equiv \{c_{pi}\}_{mr},$$

$$HH^T = \Big\{\sum_{j=1}^{n} h_{ij}h_{ji}\Big\}_{rr} \equiv \{d_{ki}\}_{rr},$$

and

$$[\text{grad}(W)]_i^T = \begin{pmatrix} \sum_{k=1}^{r} w_{ik}d_{k1} - c_{i1} \\ \sum_{k=1}^{r} w_{ik}d_{k2} - c_{i2} \\ \cdots \\ \sum_{k=1}^{r} w_{ik}d_{kr} - c_{ir} \end{pmatrix} = \begin{pmatrix} B_1 \\ B_2 \\ \cdots \\ B_r \end{pmatrix}.$$

From (9), it follows that

$$\xi_i^* = \frac{\sum_{i=1}^{r}(B_i)^2}{\sum_{j=1}^{n}[\sum_{i=1}^{r}(h_{ji}B_i)]^2}.$$

If $\xi_i^* > \frac{w_{ij}}{[\text{grad}(W)]_{ij}}$, then $\xi_{ij} = \frac{w_{ij}}{[\text{grad}(W)]_{ij}}$. From (11), for the $j$-th element of $W_i^T$, there is

$$w_{ij} \longleftarrow w_{ij} - \frac{w_{ij}}{[\text{grad}(W)]_{ij}}[\text{grad}(W)]_{ji}^T.$$

Then, we have

$$w_{ij} \longleftarrow w_{ij} - \frac{w_{ij}}{\sum_{k=1}^{r} w_{ik}d_{kj} - c_{ij}}(\sum_{k=1}^{r} w_{ik}d_{kj} - c_{ij}) = 0.$$

If $\xi_i^* \leq \frac{w_{ij}}{[\text{grad}(W)]_{ij}}$, then $\xi_{ij} = \xi_i^* = \frac{\|[\text{grad}(W)]_i^T\|^2}{\|H^T[\text{grad}(W)]_i^T\|^2}$. Similarly, the learning rule will be

$$w_{ij} \longleftarrow w_{ij} - \frac{\|[\text{grad}(W)]_i^T\|^2}{\|H^T[\text{grad}(W)]_i^T\|^2}[\text{grad}(W)]_{ji}^T.$$

Hence, we need to discuss the convergence of the following algorithm,

$$w_{ij} \longleftarrow w_{ij} - B_i \frac{\sum_{i=1}^{r}(B_i)^2}{\sum_{j=1}^{n}[\sum_{i=1}^{r}(h_{ji}B_i)]^2}. \tag{46}$$

**Definition 2.** *For the algorithm (46), a point $w_{ij} \in R$ is called an equilibrium if and only if it satisfies*

$$B_i \frac{\sum_{i=1}^{r}(B_i)^2}{\sum_{j=1}^{n}[\sum_{i=1}^{r}(h_{ji}B_i)]^2} = 0. \tag{47}$$

Denote

$$c_1 = \min_{1\leq p\leq m, 1\leq i\leq r}\{c_{pi}\},\ c_2 = \max_{1\leq p\leq m, 1\leq i\leq r}\{c_{pi}\},$$

$$d_1 = \min_{1\leq k,i\leq r}\{d_{ki}\},\ d_2 = \max_{1\leq k,i\leq r}\{d_{ki}\},$$

$$h_1 = \min_{1\leq i\leq r, 1\leq j\leq n}\{h_{ij}\},\ h_2 = \max_{1\leq i\leq r, 1\leq j\leq n}\{h_{ij}\},$$

$$\sum_{k=1}^{r} w_{ik} = \sum_{q=1}^{j-1} w_{qk} + \sum_{q=j+1}^{r} w_{qk} + w_{ij} = Q + w_{ij}.$$

From (22), contrarily, only $w_{ij}$ is variable and all the others are constants, then we have

$$d_1(Q + w_{ij}) - c_2 \leq B_i \leq d_2(Q + w_{ij}) - c_1, \tag{48}$$

which is similar to the inequality (29). From this expression, we have the following Theorem.

**Theorem 4.** *There exists a non-zero equilibrium point of algorithm (46), for some constants $0 \leq \eta_1 \leq 1$ and $\eta_2 \geq 1$, the structure of equilibrium point is $w_{ij} = \frac{c_1}{\eta_1 d_2} - Q$ or $w_{ij} = \frac{c_2}{\eta_2 d_1} - Q$ in the condition $\eta_2 c_1 d_1 = \eta_1 c_2 d_2$.*

**Proof.** The proof is similar to Theorem 2 and Theorem 3. □

## IV. Invariant Sets

Using the similar method in [16], we construct the invariant sets to constrain the area of the initializations.

**Definition 3.** [21] *A set S is called an invariant set for a dynamic system $x_{n+1} = f(x_n)$, if for any initial value $x_0 \in S$, the updates $x_{n+1}$ of system starting from $x_0$ will remain in $S$ for all $n \geq 0$.*

From the proof of Theorem 3, we assume all the others are constants except $h_{ij}$, which can be decided in the upcoming discussion. We can prove Theorems 5 and 6.

Denote

$$H_1 = \{h_{ij} | h_{ij} \in R,\ \frac{a_1}{b_2} - P \leq h_{ij} \leq \frac{a_2}{b_1} - P\}, \quad (49)$$

$$W_1 = \{w_{ij} | w_{ij} \in R,\ \frac{c_1}{d_2} - Q \leq w_{ij} \leq \frac{c_2}{d_1} - Q\}. \quad (50)$$

**Theorem 5.** *Suppose $a_1$, $a_2$, $b_1$, $b_2$ and $P$ are constants, $H_1$ is an invariant set of algorithm (17).*

**Proof.** Suppose

$$0 < \frac{a_1}{b_2} - P \leq h_{ij}(t) \leq \frac{a_2}{b_1} - P. \quad (51)$$

From algorithm (17), for the $(t+1)$th update, we have

$$h_{ij}(t+1) = h_{ij}(t) - A_i \frac{\sum_{i=1}^{r} (A_i)^2}{\sum_{p=1}^{m} [\sum_{i=1}^{r} (w_{pi} A_i)]^2}. \quad (52)$$

From the proof of Theorem 3, we need to discuss two cases. For case 1, from the inequality (34), it follows that

$$h_{ij}(t) - \frac{[b_2(P+h_{ij}(t)) - a_1]^3}{mrw_1^2[b_1(P+h_{ij}(t)) - a_2]^2} \leq h_{ij}(t+1) \leq h_{ij}(t) - \frac{[b_1(P+h_{ij}(t)) - a_2]^3}{mrw_2^2[b_2(P+h_{ij}(t)) - a_1]^2}, \quad (53)$$

where $mw_1^2 = b_1$, $mw_2^2 = b_2$, $a_1$, $a_2$, $b_1$, $b_2$ and $P$ are constants. From (32), we have $\mid b_2(P + h_{ij}) - a_1 \mid \geq \mid b_1(P + h_{ij}) - a_2 \mid$. Thus, from (52), there is

$$h_{ij}(t) - \frac{b_2(P + h_{ij}(t)) - a_1}{rb_1} \times 1 \leq h_{ij}(t+1) \leq h_{ij}(t) - \frac{b_1(P + h_{ij}(t)) - a_2}{rb_2} \times 1. \quad (54)$$

The above inequality follows that

$$\frac{(rb_1 - b_2)h_{ij}(t) - b_2P + a_1}{rb_1} \leq h_{ij}(t+1) \leq \frac{(rb_2 - b_1)h_{ij}(t) - b_1P + a_2}{rb_2}.$$

From the inequality (51), we have

$$\frac{(rb_1 - b_2)(\frac{a_1}{b_2} - P) - b_2P + a_1}{rb_1} \leq h_{ij}(t+1) \leq \frac{(rb_2 - b_1)(\frac{a_2}{b_1} - P) - b_1P + a_2}{rb_2}. \quad (55)$$

Thus, the above inequality can be simplified to

$$0 < \frac{a_1}{b_2} - P \leq h_{ij}(t+1) \leq \frac{a_2}{b_1} - P, \quad (56)$$

which is same as the inequality (51). The inequality (56) shows that for any $t \geq 0$, if $\frac{a_1}{b_2} - P \leq h_{ij}(t) < \frac{a_2}{b_1} - P$, there always exists $\frac{a_1}{b_2} - P \leq h_{ij}(t+1) \leq \frac{a_2}{b_1} - P$.

For case 2, from (33) and (40), using the above method, it follows that

$$h_{ij}(t) - \frac{a_2 - b_1(P + h_{ij}(t))}{rb_1} \leq h_{ij}(t+1) \leq h_{ij}(t) - \frac{a_1 - b_2(P + h_{ij}(t))}{rb_2}.$$

Similarly, from the inequality (51), the above inequality becomes

$$0 < \frac{a_1}{b_2} - P \leq h_{ij}(t+1) \leq \frac{a_2}{b_1} - P,$$

which is same as the inequality (56).

Therefore,$H_1$ is an invariant set of algorithm (17). □

**Theorem 6.** *Suppose $c_1$, $c_2$, $d_1$, $d_2$ and $Q$ are constants, $W_1$ is an invariant set of algorithm (46).*

**Proof.** The proof can be omitted since it's similar to Theorem 5.

Theorem 5 and 6 guarantee that any trajectory of algorithm (17) and algorithm (46) starting from any points in the invariant sets $H_1$ and $W_1$ will stay in $H_1$ and $W_1$ correspondingly. For some initializations are not in the invariant sets, the $W$ and $H$ are approaching to be negative in the iterations, which will be verified in the experiments. In the following section, we will prove the equilibrium point is stable to complete the proof of convergence.

## V. Convergence Analysis

**Theorem 7.** *For the algorithm (17), the non-zero equilibrium point is stable if it satisfies the conditions (45) and (49).*

**Proof.** From (52) and (22), it follows that

$$h_{ij}(t+1) = h_{ij}(t) - f(h_{ij}(t)). \quad (57)$$

To discuss the convergence of algorithm (17), from (18), we just need to discuss the stability of $f(h_{ij})$. From the proof of Theorem 3, we discuss two cases to obtain the structure of equilibrium point. Thus, we should also discuss two cases to prove the stability of equilibrium point.

For case 1, from the inequality (35), for simplicity, we denote

$$G_1(h_{ij}) = \frac{[\eta_1 b_2(P + h_{ij}) - a_1]^3}{mrw_1^2[b_1(P + h_{ij}) - a_2]^2} \quad (58)$$

and

$$G_2(h_{ij}) = \frac{[\eta_2 b_1(P + h_{ij}) - a_2]^3}{mrw_2^2[b_2(P + h_{ij}) - a_1]^2}, \quad (59)$$

then we have

$$G_2(h_{ij}) \leq f(h_{ij}) \leq G_1(h_{ij}).$$

Using Lemma 1 to obtain the equilibrium point, $G_1(h_{ij})$ and $G_2(h_{ij})$ must satisfy monotonicity. In addition, the terms that we substitute the equilibrium point into the derivatives of $G_1(h_{ij})$ and $G_2(h_{ij})$ need to be less than 1 [21].

First, we discuss the following derivatives of $G_1(h_{ij})$ and $G_2(h_{ij})$.

$$\begin{aligned}&G_1'(h_{ij})\\&=\frac{[\eta_1 b_2(P+h_{ij})-a_1]^2[b_1(P+h_{ij})-a_2]}{mrw_1^2[b_1(P+h_{ij})-a_2]^4}\times\\&\{3\eta_1 b_2[b_1(P+h_{ij})-a_2]-2b_1[\eta_1 b_2(P+h_{ij})-a_1]\},\end{aligned}$$

similarly

$$\begin{aligned}&G_2'(h_{ij})\\&=\frac{[\eta_2 b_1(P+h_{ij})-a_2]^2[b_2(P+h_{ij})-a_1]}{mrw_2^2[b_2(P+h_{ij})-a_1]^4}\times\\&\{3\eta_1 2b_1[b_2(P+h_{ij})-a_1]-2b_2[\eta_2 b_1(P+h_{ij})-a_2]\}.\end{aligned}$$

Substituting the non-zero solutions of (38) and (39) into the above equations, we obtain

$$\frac{d(G_1(h_{ij}))}{dh_{ij}}|\frac{a_1}{\eta_1 b_2}-P=0$$

and

$$\frac{d(G_2(h_{ij}))}{dh_{ij}}|\frac{a_2}{\eta_2 b_1}-P=0.$$

Hence, there is

$$\frac{d(G_1(h_{ij}))}{dh_{ij}}|\frac{a_1}{\eta_1 b_2}-P=\frac{d(G_2(h_{ij}))}{dh_{ij}}|\frac{a_2}{\eta_2 b_1}-P<1,$$

which means since any time when $h_{ij}(t)=\frac{a_1}{\eta_1 b_2}-P$ or $h_{ij}(t)=\frac{a_2}{\eta_2 b_1}-P$, the derivatives of the terms $f(h_{ij}(t+1))$, $f(h_{ij}(t+2)),\ldots$ will always be less than 1. It satisfies the condition of stability. Therefore, the equilibrium point is stable if $G_1(h_{ij})$ and $G_2(h_{ij})$ are monotonous.

Then, we discuss the monotonicity of $G_1(h_{ij})$ and $G_2(h_{ij})$. To simplify the analysis, we denote

$$G(x)=\frac{[\eta c_1(P+x)-c_2]^3}{mrc_3^2[c_4(P+x)-c_5]^2},\tag{60}$$

where $c_1$, $c_2$, $c_4$, $c_5$ and $c_3$ represent the same position of $b_2$, $a_1$, $b_1$, $a_2$ and $w_1$ in (58), similarly, $c_1$, $c_2$, $c_4$, $c_5$ and $c_3$ represent the same position of $b_1$, $a_2$, $b_2$, $a_1$ and $w_2$ in (59). The derivative of $G(x)$ is

$$\begin{aligned}G'(x)\quad&=[\eta c_1(P+x)-c_2]^2[c_4(P+x)-c_5]\times\\&\frac{[\eta c_1 c_4(P+x)-3\eta c_1 c_5+2c_2 c_4]}{mrc_3^2[c_4(P+x)-c_5]^4}.\end{aligned}$$

If $G'(x)\geq 0$, then $G(x)$ is monotonous. Hence, we need

$$c_4(P+x)-c_5\geq 0 \text{ and } \eta c_1 c_4(P+x)-3\eta c_1 c_5+2c_2 c_4\geq 0.$$

For $x=h_{ij}\geq 0$, $c_2 c_4\geq 0$ and $P\geq 0$, the equilibrium point is stable if

$$c_4 P-c_5\geq 0\tag{61}$$

and

$$\eta c_1 c_4 P-3\eta c_1 c_5\geq 0.\tag{62}$$

To guarantee the monotonicity of $G(x)$, we need both the above two inequalities (61) and (62) are satisfied.

From the inequality (61), we need

$$b_1 P-a_2\geq 0.$$

From (45), the above inequality becomes

$$b_1 P-\frac{\eta_2 a_1 b_1}{\eta_1 b_2}\geq 0.$$

Since $0\leq\eta_1\leq 1$ and $\eta_2\geq 1$, to guarantee the above inequality holds, we need

$$b_2 P\geq\eta_1 b_2 P\geq\eta_2 a_1\geq a_1.$$

Hence, to guarantee (61) holds, the following inequality must be satisfied.

$$b_2 P\geq a_1.\tag{63}$$

Similarly, from the inequality (62), we need

$$\eta_1 b_1 b_2 P-3\eta_2 b_2 a_2\geq 0.$$

From (45), the above inequality becomes

$$\eta_1^2 b_1 b_2 P-3\eta_2^2 a_1 b_1\geq 0.$$

For $0\leq\eta_1\leq 1$ and $\eta_2\geq 1$, to guarantee the above inequality holds, we need

$$b_2 P\geq\eta_1^2 b_2 P\geq 3\eta_2^2 a_1\geq a_1.$$

Hence, to guarantee (62) holds, the following inequality must be satisfied,

$$b_2 P\geq a_1,\tag{64}$$

For case 2, by using the above method, from the inequality (41), denote

$$G_1(h_{ij})=\frac{[\eta_1 b_2(P+h_{ij})-a_1]^2}{mrw_2^2[b_1(P+h_{ij})-a_2]}$$

and

$$G_2(h_{ij})=\frac{[\eta_2 b_1(P+h_{ij})-a_2]^2}{mrw_1^2[b_2(P+h_{ij})-a_1]}.$$

Then, we obtain

$$\frac{d(G_1(h_{ij}))}{dh_{ij}}|\frac{a_1}{\eta_1 b_2}-P=\frac{d(G_2(h_{ij}))}{dh_{ij}}|\frac{a_2}{\eta_2 b_1}-P<1.$$

Using the same way, from (60), we denote

$$G(x)=\frac{[\eta c_1(P+x)-c_2]^2}{mrc_3^2[c_4(P+x)-c_5]},$$

and the derivative of $G(x)$ is

$$G'(x)=\frac{[\eta c_1(P+x)-c_2][\eta c_1 c_4(P+x)-2\eta c_1 c_5+c_2 c_4]}{mrc_3^2[c_4(P+x)-c_5]^2}.$$

By using the similarly mathematical analysis, there exists $G'(x)\geq 0$ in the condition $b_2 P\geq a_1$ which is same as the inequality (63). Fortunately, under the condition (45), the inequality $b_2 P\geq a_1$ holds. □

By the same way, we have the stability theorem for $w_{ij}$ in algorithm (46).

**Theorem 8.** *For the algorithm (46), under the condition (50), the non-zero equilibrium point exists and is stable.*

From Theorem 3 and Theorem 7, we have that the equilibrium point of algorithm (17) is stable, and from Theorem 5 we know that all the initializations are constrained in the invariant sets, which means the equilibrium point of algorithm (17) is stable in $H_1$.

According to the Theorem 5 in [22], we obtain the cAG algorithm is locally convergent at the equilibrium points in the invariant sets. There are two conditions to constrain the convergence of the cAG algorithm. For $h_{ij}$, one condition comes from the monotonicity constraints of (58) and (59), and another condition comes from the restrictions on invariant set $H_1$. For some initializations which are not in the invariant sets, the W and H are approaching to be negative in the iterations. Now, we will confirm this circumstance in the experiments.

## VI. Simulation and Discussions

NMF algorithms have been extensively used to test data for signals and images with various statistical distributions. In this section, first, we present numerical experimental results to illustrate the convergence of the cAG algorithm and analyze the importance of the two convergence conditions, especially the invariant sets. Second, by using the same way, the convergence conditions of the MU algorithm are obtained. We compare the cAG algorithm with the MU algorithm by analyzing the difference of the convergence conditions and theoretically illustrate the advantages of the cAG algorithm.

### A. Convergence of the cAG Algorithm in Experiments.

By testing several groups data, we want to choose suitable initializations to obtain the correct data separation for NMF algorithms in the practical application. To estimate the original sources in a very small error, the stable conditions (50), (51) and (65) are satisfied in the initializations. Similar as [16], we have the following description of the algorithm steps.

1. Initialize the vector $h_{ij}$ and non-negative matrix $W = \{w_{pi}\}_{mr}$ with positive decimal numbers;
2. According the computing result of $P$, $Q$ and $\{v_{pj}\}_{m\times n}$, select suitable $\alpha = 1 - \frac{w_2^2}{w_1}$ for the test;
3. To avoid the divergence, set the threshold for $h_{ij}$ and $w_{ij}$ to 0.001-0.005, testing the convergence condition;
4. Compute $h_{ij}$ and $w_{ij}$ according to the algorithm (17) and (46) to obtain the factorization of $v_{pj}$.
5. If $h_{ij}$ and $w_{ij}$ have not converged, go back to step 3.

In experiment 1, we first select a group of data to demonstrate the analysis results. Without loss of generality, we only consider the convergence of $w_{11}$ and $h_{11}$ with different initializations. From algorithm (17), it follows that

$$h_{11} \longleftarrow h_{11} - A_1 \frac{\sum_{i=1}^{r}(A_i)^2}{\sum_{p=1}^{m}[\sum_{i=1}^{r}(w_{pi}A_i)]^2}, \qquad (65)$$

where $A_i = \sum_{k=1}^{r} b_{ik}h_{k1} - a_{i1}$, $b_{ik} = \sum_{p=1}^{m} w_{ip}w_{pk}$ and $a_{i1} = \sum_{p=1}^{m} w_{ip}v_{p1}$.

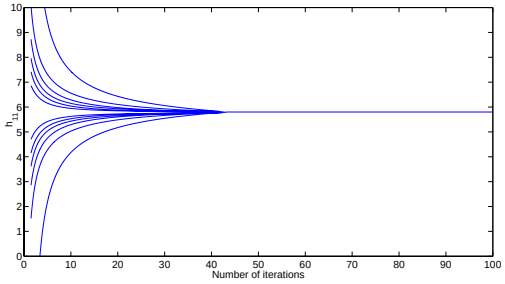


Fig. 1 The convergence of $h_{11}$.

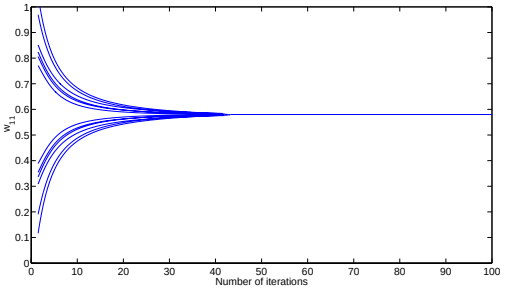


Fig. 2 The convergence of $w_{11}$.

We test the algorithm (65) for a randomly generated full column rank matrix $V \in R^{100\times 100}$. For initializations of $P = \sum_{k=1}^{r} h_{k1} - h_{11} = 1.397$, with the condition $b_2P \geq a_1$, where $b_2 = \sum_{p=1}^{m} w_2w_2 = mw_2^2$, and $a_1 = \sum_{p=1}^{m} w_1v_{pj} = mw_1v_{pj}$, we select $\alpha = 0.01$. Then, there is $\frac{w_2^2}{w_1} = 1 - \alpha = 1 - 0.01 = 0.99$. Here, we show the convergence of $h_{11}$ in Fig. 1 and the convergence of $w_{11}$ in Fig. 2, respectively. Therefore, for different initializations of $h_{11}$, it always converges to the same constant if the condition $b_2P \geq a_1$ is satisfied. And $w_{11}$ has the same result. The convergence of the cAG algorithm is verified.

In experiment 2, to illustrate the importance of the invariant sets in the cAG algorithm, we test the algorithm (65) for a randomly generated full column rank matrix $V$ and the matrix $W \in R^{100\times 100}$. For initializations of $P = 67.19$, with the condition $b_2P \geq a_1$, we select $w_1 = 0.1$ and $w_2 = 1.4$. To show the simulation result, we change $v_{11}$.

Fig. 3 shows the importance of the initializations in the cAG algorithm. A good initialization can improve the speed and accuracy of the algorithm. If the initializations are not in the invariant sets, even it satisfies the condition (63), the $W$ and $H$ are approaching to be negative in the iterations. It's clear that the negative matrix separation is meaningless.

We verify the convergence of the cAG algorithm and illustrate the importance of the initializations in experiments. In addition, the conditions of the initialization constraints can accelerate the convergence of the algorithm. It will lay a foundation for better understanding of the cAG algorithm.

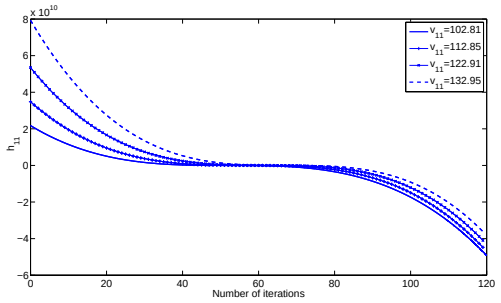


Fig. 3 The initializations are not in the invariant sets.

### B. Comparison of Algorithms

In this subsection, we analyze the convergence of the MU algorithm by using the same method. From (3) and (4), we obtain the following two algorithms,

$$h_{ij} \longleftarrow \frac{h_{ij}a_{ij}}{\sum_{k=1}^{r} b_{ik}h_{kj}} \tag{66}$$

and

$$w_{ij} \longleftarrow \frac{w_{ij}c_{ij}}{\sum_{k=1}^{r} w_{ik}d_{kj}}. \tag{67}$$

Then the equilibrium points of algorithm (66) are

$$\Big\{0, \frac{\eta_2 a_1}{b_2} - P\Big\}, \text{and } \Big\{0, \frac{\eta_1 a_2}{b_1} - P\Big\}, \tag{68}$$

which verify the MU algorithm has zero divisor.

Considering the non-zero solutions $h_{ij} = \frac{\eta_2 a_1}{b_2} - P$ and $h_{ij} = \frac{\eta_1 a_2}{b_1} - P$, the equilibrium point is stable, if it satisfies the following conditions,

$$\eta_2 a_1 b_1 = \eta_1 a_2 b_2 \tag{69}$$

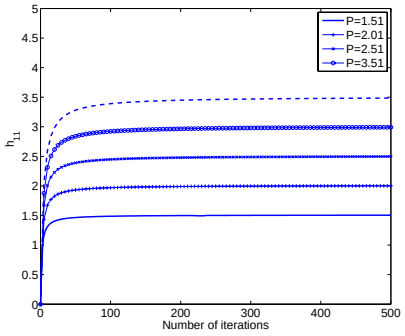


Fig. 4 In the condition $b_2P = a_1$, the algorithm (66) is stable.

and

$$b_2P < a_1. \tag{70}$$

Comparing with the inequalities (45) and (70), we want to discuss the convergence in the condition $b_2P = a_1$ in the experiments.

In experiment 3, we test the algorithm (66) for the same matrix $V \in R^{100\times100}$ as that in experiment 1. From $b_2P = a_1$, there is $\frac{w_1}{w_2^2} = \frac{P}{v}$. For different initializations of $P$, with the condition $\frac{w_1}{w_2^2} = \frac{P}{v}$, Fig. 4 shows the simulation result.

Fig. 4 shows the algorithm (66) is convergent in the condition $b_2P = a_1$. By testing several groups data, we find the results are the same. For different initializations, the MU algorithm always converges to the same constant if the condition $b_2P = a_1$ is satisfied.

Hence we obtain the non-zero equilibrium point of the MU algorithm is stable, if the following equation is satisfied

$$b_2P \le a_1. \tag{71}$$

The two inequalities (45) and (71) are of opposite sign. Then we discuss the case of zero solution.

Using the same method in Section IV, we obtain the invariant sets of MU algorithm as follows,

$$H_1 = \{h_{ij} | h_{ij} \in R,\ \frac{a_1}{b_2} \le h_{ij} \le \frac{a_2}{b_1}\},$$

and

$$W_1 = \{w_{ij} | w_{ij} \in R,\ \frac{c_1}{d_2} \le w_{ij} \le \frac{c_2}{d_1}\}.$$

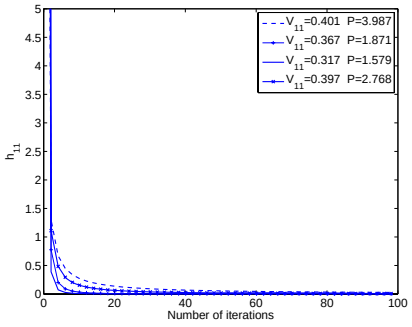


Fig. 5 The MU algorithm is convergent at zero.

Similarly, using the same method in Section V, we obtain the algorithm (66) will be convergent at zero if

$$b_2P > a_1. \tag{72}$$

It's clear that the convergence conditions of the MU algorithm at zero are similar to that of the cAG algorithm at non-zero.

In experiment 4, to illustrate the different convergence properties of the cAG algorithm and the MU algorithm in the condition $b_2P > a_1$, we test the algorithm (66) for a new generated matrix $V \in R^{100\times100}$. With different initializations of $P$ and $v_{11}$, the convergence of the MU algorithm is shown in Fig. 5.

Fig. 5 shows the MU algorithm is convergent at zero in the condition $b_2P > a_1$. However, it is not meaningful to discuss the convergence at zero. Thus, when the condition $b_2P > a_1$ is satisfied, which means $\frac{w_2^2}{w_1} > \frac{v_{pj}}{P}$, using the cAG algorithm is more effective. More concretely, when the MU algorithm is convergent at zero, to prevent validly the appearance of zero divisor, we use the cAG algorithm for NMF. Therefore, from the above experiments, we theoretically illustrate the advantages of the cAG algorithm.

## VII. Conclusion

In this paper, we analyze the convergence of the cAG algorithm by utilizing the stability theorem of the equilibrium point. We verify that the cAG algorithm has not zero divisor. The convergence conditions and the structure of the equilibrium points are also obtained. In addition, the conditions of the initialization constraints can accelerate the convergence of the algorithm. Using the same convergence analysis, we verify the MU algorithm has zero divisor and obtain the convergence conditions at zero. In the end, simulations illustrate the advantages of the cAG algorithm and confirm our theories.

## Acknowledgment

This work was supported in part by the National Natural Science Foundation of China (60702071), Program for New Century Excellent Talents in University (NCET-06-0811), 973 National Basic Research Program of China, (2010CB732501), Fundation of Sichuan Excellent Young Talents(09ZQ026-035), Open Project of State Key Lab. for Novel Software Technology of Nanjing University and Zhejiang Provincial Natural Science Foundation of China.

## References

[1] D.D. Lee, H.S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788-791.
[2] E. F. Gonzales, Y. Zhang, Accelerating the Lee-Seung algorithm for non-negative matrix factorization, Dept. Comput. Appl. Math., Rice Univ., Houston, TX, Tech. Rep. (2005).
[3] M. Berry, M. Browne, A. Langville, P. Pauca, R. Plemmons, Algorithms and applications for approximate nonnegative matrix factorization, Computational Statistics and Data Analysis 52 (2006) 155-173.
[4] P. Paatero, U. Tapper, Positive matrix factorization: a non-negative factor model with optimal utilization of error estimates of data values, Environmetrics, 5 (1994) 111-126.
[5] M. Chu, F. Diele, R. Plemmons, S. Ragni, Optimality, computation, and interpretation of nonnegative matrix factorizations, unpublished report (2004).
[6] P. Paatero and U. Tapper, Positive matrix factorization: A non-negative factor model with optimal utilization of error, Environmetrics, 5 (1994) 111-126.
[7] L.K. Saul, F. Sha, D.D. Lee, Statistical signal processing with nonnegativity constraints, Proceedings of EuroSpeech 2 (2003) 1001-1004.
[8] F. Shahnaz, M.W. Berry, V.P. Pauca, R. Plemmons, Document clusting using nonnegative matrix factorization, Information Processing and Management 42 (2006) 373-386.
[9] P.M. Kim, B. Tidor, Subsystem identification through dimensionality reduction of large-scale gene expression data Genome Research, 13 (2003) 1706C1718.
[10] V.P. Pauca, J. Piper, R.J. Plemmons, Nonnegative matrix factorization for spectral data analysis, Linear Algebra and its Applications 416 (2006) 29-47.
[11] F. Sha, L.K. Saul, Real-time pitch determination of one or more voices by nonnegative matrix factorization, Advances in Neural Information Processing Systems (2005). online papers.
[12] P. Smaragdis, J.C. Brown, Non-negative matrix factorization for polyphonic music transcription, in: Proceedings of IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (2003) 177-180.
[13] L. Lu, Alternative gradient algorithms with applications to nonnegative matrix factorizations, Applied Mathematics and Computation 216 (2010) 1763-1770.
[14] A. Cichocki, R. Zdunek, S. Amari, New algorithms for non-negative matrix factorization in applications to blind source separation, ICASSP 2006, Toulouse, France, 621-625.
[15] C.J. LIN, Projected gradient methods for non-negative matrix factorization, Tech. Report Information and Support Service ISSTECH-95-013 (2005).
[16] S.M. Yang, Z. Yi, Convergence analysis of non-negative matrix factorization for BSS algorithm, Neural Process Lett (2010) 45-64.
[17] A. Cichocki, R. Zdunek, Multilayer non-negative matrix factorization using project gradient approaches, International Journal of Neural Systems (2007) 431-446.
[18] R. Salakhutdinov, S.T. Roweis, Z. Ghahramani, The convergence of bound optimization algorithms, In: The 19th International Conference on Uncertainty in Artificial Intelligence, (2003).
[19] C.J. Lin, On the convergence of multiplicative update algorithms for nonnegative matrix factorization, Transactions on Neural Networks (2007) 18.
[20] V.L. Kocic, G. Ladas, Global behavior of nonlinear difference equations of higher order, Kluwer Academic Publishers (1953).
[21] J.K. Hale, Theory of functional differential equations, Springer, Heidelberg, New York (1977).
[22] J. Dai, S. Meyn, Stabilitity and convergence of moments for multiclass queueing networks via fluid limit models, Translation on Automatic Control (1995) 1889-1904.