Research Article

Two Efficient Algorithms for Orthogonal Nonnegative Matrix Factorization

Jing Wu¹,^{ID} Bin Chen,² and Tao Han³

¹School of Science, Xijing University, Xi'an 710123, China

²School of Mathematics and Statistics, Weinan Normal University, Weinan 714099, China

³School of Science, Xi'an University of Technology, Xi'an 710048, China

Correspondence should be addressed to Jing Wu; 656125381@qq.com

Received 11 August 2020; Accepted 8 June 2021; Published 1 July 2021

Academic Editor: Predrag S. Stanimirović

Copyright © 2021 Jing Wu et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Nonnegative matrix factorization (NMF) is a popular method for the multivariate analysis of nonnegative data. It involves decomposing a data matrix into a product of two factor matrices with all entries restricted to being nonnegative. Orthogonal nonnegative matrix factorization (ONMF) has been introduced recently. This method has demonstrated remarkable performance in clustering tasks, such as gene expression classification. In this study, we introduce two convergence methods for solving ONMF. First, we design a convergent orthogonal algorithm based on the Lagrange multiplier method. Second, we propose an approach that is based on the alternating direction method. Finally, we demonstrate that the two proposed approaches tend to deliver higher-quality solutions and perform better in clustering tasks compared with a state-of-the-art ONMF.

1. Introduction

Nonnegative matrix factorization (NMF) has been investigated by many researchers, such as Paatero and Tapper [1]. However, it was only through the works of Lee and Seung published in Nature and NIPS [2, 3] that this method has gained popularity. Based on the argument that nonnegativity is crucial for human perception, they proposed a simple algorithm to find nonnegative representations of nonnegative data and images. The basic NMF is a technique that decomposes a nonnegative data matrix into a pair of other nonnegative matrices with lower ranks:

$$Y \approx AX, \quad (1)$$

where $Y \in \mathbb{R}_+^{I \times T}$ denotes a nonnegative data matrix. $A \in \mathbb{R}_+^{I \times J}$ and $X \in \mathbb{R}_+^{J \times T}$ denote the basis matrix and coefficient matrix, respectively. J denotes the number of factors that is usually chosen so that $J \ll \min(I, T)$. Recently, NMF has proven to be useful for many applications in pattern recognition, multimedia, text mining, and clustering, as well as environment and DNA gene expression classifications [4–12].

In solving the NMF problem, the update rules given by Lee and Seung [3] take a multiplicative form and satisfy the nonnegative constraint implicitly and elegantly. The extension of the multiplicative updates provided by them [3] can be found, for instance, in the works of Sha et al. [10], whereby a multiplicative update rule is proposed to solve a nonnegative quadratic programming problem. However, the slow convergence of the multiplicative updating rule has been pointed out, and more efficient algorithms equipped with more powerful theoretical convergence properties have been introduced. These efficient algorithms are based on either the alternating nonnegative least squares (ANLS) framework or the hierarchical alternating least squares (HALS) method.

Recently, other forms of update rules have been studied to increase the speed of convergence. For instance, a quasi-Newton method for NMF was considered by Zdunek and Cichocki [13], and a projected gradient method was proposed by Hoyer and Lin [6, 14].

There has also been significant interest in orthogonal nonnegative matrix factorization (ONMF), which imposes orthogonality constraints on the factor matrix [15–18]. It has

been proven that ONMF is equivalent to K-means clustering [16, 18]. Moreover, as a result of the nonnegativity and orthogonality constraints, the orthogonal factor matrix is naturally sparse. In such cases, the factorization is essentially unique, and we can ignore the permutation problem.

Ding et al. and Pompili et al. [16, 18] proposed two different kinds of methods to solve the ONMF problem. In a study conducted by Ding et al. [16], ONMF algorithms strictly enforce nonnegativity for each iterate while trying to achieve orthogonality at the limit. This can be done using a proper penalization term [16], a projection matrix formulation [19], or choosing a suitable search direction [15]. In a study conducted by Pompili et al. [18], the authors proposed a method working the opposite way: at each iteration, a projected gradient scheme is used to ensure that the orthogonal factor iterates are orthogonal but not necessarily nonnegative. However, these algorithms still are not convergent algorithms for ONMF. In this study, we first briefly introduce NMF and ONMF algorithms. Next, based on the Lagrange multiplier, we design a convergence algorithm, which is a method similar to the algorithm presented by Lin [20]. Because NMF algorithms based on the alternating direction method are more efficient than multiplicative update algorithms, we propose another convergence algorithm for solving the ONMF problem by combining the ADM approach with our convergence algorithm. Experiments on two grayscale images and the ALLAML gene-sample data demonstrate that Algorithms 2 and 3 tend to deliver higher -quality solutions and perform better in a clustering task compared with Algorithm 1.

The remainder of this article is organized as follows: Section 2 gives a brief review of NMF and ONMF algorithms. In Section 3, an improved version of the original ONMF algorithm and the convergence analysis is provided and another new ONMF algorithm based on the alternating least squares (ALS) approach is provided. Section 4 is devoted to numerical experiments. Some concluding remarks are provided in Section 5.

2. Related Algorithms

In this section, we provide a brief review of NMF and ONMF algorithms.

2.1. Nonnegative Matrix Factorization. NMF aims to find a nonnegative matrix $Y = [y_1, y_2, \dots, y_r] \in \mathbb{R}_+^{m \times r}$ and another nonnegative matrix $X^T = B = [b_1, b_2, \dots, b_r] \in \mathbb{R}_+^{n \times r}$ whose product approximate a given nonnegative matrix $Y \in \mathbb{R}_+^{m \times n}$:

$$\min_{A \geq 0, X \geq 0} \|Y - AB^T\|_F^2. \quad (2)$$

Because the NMF problem is not convex in both A and B , various algorithms have been proposed. The first method for solving the equation (2) is the ALS algorithm utilized by Paatero and Tapper in 1994. It minimizes the least squares

cost function with respect to either A or B , one at a time, by fixing the other and disregarding nonnegativity; after each least squares step, the algorithm sets all negative entries to zero. The scheme can be updated as follows:

$$A \leftarrow YX^T(XX^T)^\dagger, \quad (3)$$

$$X \leftarrow (A^T A)^\dagger A^T Y, \quad (4)$$

where $A^\dagger$ denotes the Moore–Penrose pseudoinverse.

Another popular method for NMF is the multiplicative updating method proposed by Lee and Seung. It can be expressed as follows:

$$A \leftarrow A \odot \left[\frac{YX^T \oslash (AXX^T)}{1} \right], \quad (5)$$

$$X \leftarrow X \odot \left[\frac{A^T Y \oslash (A^T A X)}{1} \right], \quad (6)$$

where $\odot$ and $\oslash$ denote elementwise multiplication and division, respectively. When they are started from non-negative initial estimates, iterates will remain nonnegative throughout the iterations. By all indications, this algorithm appears to be the most widely used solution method in NMF by far.

The modified update rule given in equations (7) and (8) was first proposed by Gillis and Glineur [21]; it can be expressed as follows:

$$A \leftarrow \max\{\varepsilon, A \odot \left[\frac{YX^T \oslash (AXX^T)}{1} \right]\}, \quad (7)$$

$$X \leftarrow \max\{\varepsilon, X \odot \left[\frac{A^T Y \oslash (A^T A X)}{1} \right]\}, \quad (8)$$

where $\varepsilon > 0$ denotes a small number. They proved that if a sequence of solutions generated by (7) and (8) has a limit point, then this point is necessarily a stationary point of the following optimization problem:

$$\min_{A \geq \varepsilon, X \geq \varepsilon} \|Y - AB^T\|_F^2. \quad (9)$$

2.2. Original ONMF Algorithms. In an ONMF, an additional orthogonal constraint, $XX^T = I$, is imposed on the NMF. We briefly review the first ONMF algorithm proposed by Ding et al. [16] and reveal the problem behind ONMF.

The goal of ONMF is to find a nonnegative orthogonal matrix X and a nonnegative matrix A by minimizing the following objective function using a Lagrangian multiplier λ . Because it is not easy to solve this problem for every value of λ , Ding et al. [16] ignored the nonnegativity and relied on $XX^T = I$ to obtain a unique value of Λ_X :

$$L(A, X) = \frac{1}{2} \|Y - AX\|_F^2 + \frac{1}{2} \text{tr}(\Lambda_X(XX^T - I)). \quad (10)$$

The KKT conditions for the objective in equation (10) are as follows:

Input: $Y \in \mathbb{R}_+^{I \times T}$ and $J \in \mathbb{N}$.
Output: Factors $A \in \mathbb{R}_+^{I \times J}$, $X \in \mathbb{R}_+^{J \times T}$, which solve problem (18).
(1) Select initial matrix $A^{(0)} > 0$ and $X^{(0)} > 0$.
(2) For $k = 1 : \text{maxIt}$
(2.1) Calculate the factor matrix A using the following formula:

$$a_{ij} \leftarrow a_{ij} - (\bar{a}_{ij} / [AXX^T]_{ij} + \delta) [\nabla_A D_F(A)]_{ij}$$

(2.2) Calculate the factor matrix X using the following formula:

$$x_{jt} \leftarrow x_{jt} - (\bar{x}_{jt} / [A^T AX + \alpha XX^T X]_{jt} + \delta) [\nabla_X D_F(X)]_{jt}$$

End Do

ALGORITHM 1: AUR algorithm for the ONMF.

Input: $Y \in \mathbb{R}_+^{I \times T}$ and $J \in \mathbb{N}$.
Output: $A \in \mathbb{R}_+^{I \times J}$, $X \in \mathbb{R}_+^{J \times T}$, which solve the optimization problem (1.1).
(1) Select initial matrices $A^{(0)} > 0$ and $X^{(0)} > 0$.
(2) For $k = 1 : \text{maxIt}$
(2.1) Calculate the factor matrix A using the following formula:

$$a_{ij} \leftarrow \max \{ a_{ij} - (\bar{a}_{ij} / [AXX^T]_{ij} + \delta) [\nabla_A D_F(A)]_{ij}, \varepsilon \}$$

(2.2) Calculate the factor matrix X using the following formula:

$$x_{jt} \leftarrow \max \{ x_{jt} - (\bar{x}_{jt} / [A^T AX + \alpha XX^T X + \Lambda Y]_{jt} + \delta) [\nabla_X D_F(X)]_{jt}, \varepsilon \}$$

(2.3) Calculate the factor matrix Λ using the following formula:

$$\Lambda \leftarrow \Lambda + \gamma \rho (XX^T - I)$$

End Do

ALGORITHM 2: Modified AUR algorithm of ONMF.

Input: $Y \in \mathbb{R}_+^{I \times T}$ and $J \in \mathbb{N}$.
Output: $A \in \mathbb{R}_+^{I \times J}$, $X \in \mathbb{R}_+^{J \times T}$, which solve the optimization problem (1.1).
(1) Select initial matrices $A^{(0)} > 0$ and $X^{(0)} > 0$.
(2) For $k = 1 : \text{maxIt}$
Update $(A_k, X_k, U_k, V_k, \Lambda_k, \Pi_k)$ by formulas (62)–(67)
If a stopping criterion is met, then
exit and output (A_k, X_k)
end if
End Do

ALGORITHM 3: ADM-based algorithm of ONMF.

$$\begin{aligned}
A^* &\geq 0, \\
X^* &\geq 0, \\
\nabla_A(A^*) &\geq 0, \\
\nabla_X(X^*) &\geq 0, \\
\nabla_A(A^*) \odot A^* &= 0, \\
\nabla_X(X^*) \odot X^* &= 0,
\end{aligned} \tag{11}$$

with

$$\begin{aligned}
\nabla_A(A) &= AXX^T - YX^T, \\
\nabla_X(X) &= A^T AX - A^T Y + \Lambda_X X.
\end{aligned} \tag{12}$$

Using the multiplicative updating method, the update rules of A and X are derived as follows:

$$\begin{aligned}
a_{ij} &\leftarrow a_{ij} \frac{[YX^T]_{ij}}{[AXX^T]_{ij}}, \\
x_{jt} &\leftarrow x_{jt} \frac{[A^T Y]_{jt}}{[(A^T A + \Lambda_X)X]_{jt}}.
\end{aligned} \tag{13}$$

The problem with this algorithm is that it is difficult to determine Λ_c . By summing over the index i , the authors found an exact formulation for the diagonal entries:

$$(\Lambda_c)_{ii} = (A^T Y X^T - A^T A)_{ii}. \quad (14)$$

The off-diagonal entries are obtained by ignoring the nonnegativity constraint on X and setting $\nabla_X(X)$ to a zero matrix:

$$\nabla_X(X) = A^T A X - A^T Y + \Lambda_X X = 0. \quad (15)$$

$$(\Lambda_X)_{ij} = (A^T Y X^T - A^T A)_{ij}, \quad \forall i \neq j. \quad (16)$$

Equation (15) is derived from equation (10) using the fact that $\|X\|_F^2 = \text{tr}(X^T X)$ and from equation (16) using the orthogonality constraint $XX^T = I$. By combining equations (14) and (16), Λ_X can be defined as follows:

$$\Lambda_c = A^T Y X^T - A^T A. \quad (17)$$

Note that the formulation with the specific value of Λ_X does not strictly satisfy orthogonality. Therefore, ONMF algorithms prioritize the approximation while relaxing the degree of orthogonality.

To improve the approximation, Mirzal [22] proposed the following cost function:

$$\begin{aligned} a_{ij} &\leftarrow a_{ij} - \frac{a_{ij}}{[AXX^T]_{ij}} [\nabla_A D_F(A)]_{ij}, \\ x_{jt} &\leftarrow x_{jt} - \frac{x_{jt}}{[A^T AX + \alpha XX^T X]_{jt}} [\nabla_X D_F(X)]_{jt}, \end{aligned} \quad (20)$$

where $[A]_{ij} = a_{ij}$ denotes the ij th entry of matrix A . Additionally, the author proposed an equally robust AUR algorithm (see Algorithm 1) for equation (18), where $\max \text{It}$

$$\min_{A, X} D_F(Y \| AX) = \frac{1}{2} \|Y - AX\|_F^2 + \frac{\alpha}{2} \|XX^T - I\|_F^2. \quad (18)$$

Using the same strategy as in the aforementioned algorithm, the update forms are derived as follows:

$$\begin{aligned} a_{ij} &\leftarrow a_{ij} \frac{[YX^T]_{ij}}{[AXX^T]_{ij}}, \\ x_{jt} &\leftarrow x_{jt} \frac{[A^T Y + \alpha X]_{jt}}{[A^T AX + \alpha XX^T X]_{jt}}. \end{aligned} \quad (19)$$

Next, the author proposed an equivalent algorithm with a robust convergence guaranteed by (1) transforming the algorithm into a corresponding AUR algorithm and (2) replacing every zero entry that does not satisfy the KKT conditions with a small positive number to prevent zero locking. This strategy was employed to derive a convergent algorithm for the ONMF.

The AUR version of equation (18) can be expressed as follows:

denotes a limit maxtime of iteration, $\nabla_A D_F$ represents the gradient of the objective function (18), and

$$\bar{a}_{ij}^k = \begin{cases} a_{ij}^k, & \text{if } \nabla_A(A^{(k)}, X^{(k)})_{ij} \geq 0, \\ \max(a_{ij}^k, \delta), & \text{if } \nabla_A(A^{(k)}, X^{(k)})_{ij} < 0, \end{cases} \quad \bar{x}_{jt}^k = \begin{cases} x_{jt}^k, & \text{if } \nabla_X(A^{(k+1)}, X^{(k)})_{jt} \geq 0, \\ \max(x_{jt}^k, \delta), & \text{if } \nabla_X(A^{(k+1)}, X^{(k)})_{jt} < 0. \end{cases} \quad (21)$$

The author proved the convergence of this algorithm. However, this algorithm requires a high computational cost. The algorithm that is based on the alternated application of these rules in equations (5) and (6) is not guaranteed to converge to a first-order stationary point. However, a slight modification proposed by Lin [20] achieves this property (generally speaking, MU is recast as a variable metric steepest descent method, and the step length is modified accordingly). In another study [23], the authors demonstrated through numerical experiments that the update rules in equations (7) and (8) work better than the original update rules in equations (5) and (6) in some cases. This indicates

that equations (7) and (8) are important not only from a theoretical point of view but also from a practical viewpoint. Therefore, in the next section, we propose convergence updates based on the alternated approach.

3. Alternating Direction Algorithm for ONMF

3.1. Classic ADM Approach. ADM is an algorithm that is intended to blend the decomposability of dual ascent with the superior convergence properties of the multipliers method. The algorithm solves problems in the form.

$$\begin{aligned} \min \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c, \end{aligned} \quad (22)$$

with variables $x \in \mathbb{R}^n$ and $z \in \mathbb{R}^m$, where $A \in \mathbb{R}^{p \times n}$, $B \in \mathbb{R}^{p \times m}$, and $c \in \mathbb{R}^p$. We assume that f and g are convex. The optimal value of equation (22) will be denoted by

$$p^* = \inf\{f(x) + g(z) | Ax + Bz = c\}. \quad (23)$$

As in the method of ADM, we form the augmented Lagrangian as follows:

$$\begin{aligned} L_\rho(x, z, y) = & f(x) + g(z) + y^T (Ax + Bz - c) \\ & + \frac{\rho}{2} \|Ax + Bz - c\|_2^2. \end{aligned} \quad (24)$$

ADM consists of the following iterations:

$$x^{k+1} := \arg \min_x L_\rho(x, z^k, y^k), \quad (25)$$

$$z^{k+1} := \arg \min_z L_\rho(x^{k+1}, z, y^k), \quad (26)$$

$$y^{k+1} := y^k + \gamma \rho (Ax^{k+1} + Bz^{k+1} - c), \quad (27)$$

where $\rho > 0$ and $\gamma \in (0, 1.618)$ represent a step length. The algorithm is similar to dual ascent and the method of multipliers. It consists of an x -minimization step (25), a z -minimization step (26), and a dual variable update (27). As in the method of multipliers, the dual variable update uses a step size that is equal to the augmented Lagrangian parameter ρ . Conversely, the classic augmented Lagrangian multiplier method requires a joint minimization with respect to both x and z , i.e., it replaces steps (26) and (27) with

$$(x^{k+1}, z^{k+1}) \leftarrow \arg \min_{x, z} \mathcal{L}_A(x, z, y^T), \quad (28)$$

which involves both $f(x)$ and $g(z)$ and could become much more expensive. In Section 3.2, we propose improved algorithms for ONMF and present the convergence analysis. We discuss the orthogonality constraint on the rows of X . Similar results can be derived for A .

3.2. Classic ADM Approach Extension to ONMF. First, we define the objective of the ONMF as follows:

$$\min_{A, X} L(A, X) = \frac{1}{2} \|Y - AX\|_F^2 + \frac{\alpha}{2} \|XX^T - I\|_F^2 + \Lambda \cdot (XX^T - I), \quad (29)$$

such that $A \geq 0$ and $X \geq 0$.

The AUR version of Problem (18) can be modified as follows:

$$\begin{aligned} a_{ij} &\leftarrow a_{ij} - \frac{\bar{a}_{ij}}{[AXX^T]_{ij} + \delta} [\nabla_A D_F(A)]_{ij}, \\ x_{jt} &\leftarrow x_{jt} - \frac{\bar{x}_{jt}}{[A^T AX + \alpha XX^T X + \Lambda Y]_{jt} + \delta} [\nabla_X D_F(X)]_{jt}. \end{aligned} \quad (30)$$

Inspired by the approach proposed by Gillis and Glineur [23] and Yoo and Choi [24], we employ these strategies to devise a convergent algorithm for NMF with an orthogonality constraint, which is Problem (4.1). The modified AUR version of Problem (4.1) can be modified as follows:

$$\begin{aligned} a_{ij} &\leftarrow \max \left\{ a_{ij} - \frac{\bar{a}_{ij}}{[AXX^T]_{ij} + \delta} [\nabla_A D_F(A)]_{ij}, \varepsilon \right\}, \\ x_{jt} &\leftarrow \max \left\{ x_{jt} - \frac{\bar{x}_{jt}}{[A^T AX + \alpha XX^T X + \Lambda Y]_{jt} + \delta} [\nabla_X D_F(X)]_{jt}, \varepsilon \right\}, \\ \Lambda &\leftarrow \Lambda + \gamma \rho (XX^T - I), \end{aligned} \quad (31)$$

where $\varepsilon > 0$ denotes a small number and $\rho > 0$ and $\gamma \in (0, 1.618)$ represent a step length.

$$\bar{a}_{ij}^k = \begin{cases} a_{ij}^k, & \text{if } \nabla_A(A^{(k)}, X^{(k)})_{ij} \geq 0, \\ \max(a_{ij}^k, \sigma), & \text{otherwise,} \end{cases} \quad \bar{x}_{jt}^k = \begin{cases} x_{jt}^k, & \text{if } \nabla_X(A^{(k+1)}, X^{(k)})_{jt} \geq 0, \\ \max(x_{jt}^k, \sigma), & \text{otherwise,} \end{cases} \quad (32)$$

stand for the modifications for avoiding zero locking with a small positive number σ and

$$\begin{aligned} \nabla_A(A) &= AXX^T - YX^T, \\ \nabla_X(X) &= A^T AX - A^T Y + \alpha XX^T X - \alpha X + \Lambda Y. \end{aligned} \quad (33)$$

The detailed procedure for the ALS-based version of Problem (4.1) can be seen in Algorithm 2.

For Algorithm 2, we obtain the following basic result:

Lemma 1. *For any natural number k , if $A^{(0)} \geq 0$ and $X^{(0)} \geq 0$, then $A^{(k)} > 0$ and $X^{(k)} > 0$.*

Proof. This statement for A is similar to the proof for X . Therefore, we only prove the statement for X . It is obvious for $k = 0$; hence, we only need to prove the results for $k \geq 1$. $\square$

Case 1. For $\nabla_X(A^{(k+1)}, X^{(k)})_{jt} \geq 0$,

$$\begin{aligned} x_{jt}^{k+1} &= \frac{[A^T A X^{(k)} + \alpha X^{(k)} X^{(k)T} X^{(k)} + \Lambda Y]_{jt} x_{jt}^k + \delta_X^k x_{jt}^k}{[A^T A X^{(k)} + \alpha X^{(k)} X^{(k)T} X^{(k)} + \Lambda Y]_{jt} + \delta_X^k} \\ &\quad - \frac{[A^T A X^{(k)} + \alpha X^{(k)} X^{(k)T} X^{(k)} + \Lambda Y]_{jt} x_{jt}^k}{[A^T A X^{(k)} + \alpha X^{(k)} X^{(k)T} X^{(k)} + \Lambda Y]_{jt} + \delta_X^k} + \frac{[A^{(k+1)T} Y + \alpha X^{(k)}]_{jt} x_{jt}^k}{[A^T A X^{(k)} + \alpha X^{(k)} X^{(k)T} X^{(k)} + \Lambda Y]_{jt} + \delta_X^k} \\ &= \frac{[\delta_X^k + A^{(k+1)T} Y + \alpha X^{(k)}]_{jt} x_{jt}^k}{(A^{(k+1)T} Y X^{(k)T})_{jt} + \delta_X^k}. \end{aligned} \quad (34)$$

With the nonnegativity of $A^{(k+1)T}$, X , and Y ,

$$x_{jt}^k \geq 0 \Rightarrow x_{jt}^{k+1} \geq 0. \quad (35)$$

Because

$$x_{jt} \leftarrow \max \left\{ x_{jt} - \frac{\bar{x}_{jt}}{[A^T A X + \alpha X X^T X + \Lambda Y]_{jt} + \delta} [\nabla_X D_F(X)]_{jt}, \varepsilon \right\}, \quad (36)$$

$$x_{jt}^{k+1} > 0. \quad (37)$$

Case 2. For $\nabla_X(A^{(k+1)}, X^{(k)})_{jt} < 0$,

$$x_{jt}^{k+1} = x_{jt}^k - \frac{\max(x_{ij}^k, \delta) \nabla_X(A^{(k+1)}, X^{(k)})_{jt}}{[\nabla_A D_F(A)]_{ij}^+ + \delta_X^{(k)}}. \quad (38)$$

Because $\max(x_{ij}^k, \delta) > 0$ and $\nabla_X(A^{(k+1)}, X^{(k)})_{jt} < 0$, equation (27) holds.

Similarly, the statement of A can be proven.

3.2.1. Convergence Analysis. To analyze the convergence of Algorithm 2, we first prove its nonincreasing property under the update rules in equations (25) and (26), i. e.,

$$L(A^{(k+1)}, X^{(k+1)}) \leq L(A^{(k)}, X^{(k)}). \quad (39)$$

Because the algorithm solves Problem (1.1) in an alternating method, $L(A^{(k)})$ and $L(X^{(k)})$ can be analyzed separately. Because the update rule of A is similar to that of X , it is sufficient to prove the nonincreasing property of $L(X^{(k)})$.

By making use of the auxiliary function approach [3], the nonincreasing property of $L(X^{(k)})$ can be proved by showing that

$$\begin{aligned} L(X^{(k+1)}) &\leq G(A^{(k+1)}, X^{(k+1)}) \leq G(A^{(k+1)}, X^{(k)}) \\ &\leq G(A^{(k)}, X^{(k)}) \leq L(X^{(k)}), \end{aligned} \quad (40)$$

where the auxiliary function G is defined in a similar function by [22]

$$\begin{aligned} G(\mathbf{X}, \mathbf{X}^{(k)}) &= L(\mathbf{X}^{(k)}) + \text{tr} \left\{ (\mathbf{X} - \mathbf{X}^{(k)})^T \nabla_X L(\mathbf{X}^{(k)}) \right\} \\ &\quad + \frac{1}{2} \text{tr} \left\{ (\mathbf{X} - \mathbf{X}^{(k)})^T \mathbf{D} (\mathbf{X} - \mathbf{X}^{(k)}) \right\}, \end{aligned} \quad (41)$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1 & & & \\ & \mathbf{x}_2 & & \\ & & \ddots & \\ & & & \mathbf{x}_T \end{pmatrix} \in \mathbb{R}_+^{T \times J}, \quad (42)$$

where $\mathbf{x}_i (i = 1, 2, \dots, T)$ denotes the i th column of $\mathbf{X}$. Meanwhile, we define

$$\nabla_X L(\mathbf{X}^{(k)}) = \begin{pmatrix} \nabla_X L(\mathbf{X}^{(k)})_1 & & & \\ & \nabla_X L(\mathbf{X}^{(k)})_2 & & \\ & & \ddots & \\ & & & \nabla_X L(\mathbf{X}^{(k)})_T \end{pmatrix} \in \mathbb{R}_+^{T \times J}, \quad (43)$$

where $\nabla_{\mathbf{X}} L(\mathbf{X}^{(k)})_i$ denotes the i th column of $\nabla_{\mathbf{X}} L(\mathbf{X}^{(k)})$, and

$$\mathbf{D} = \text{diag}(D^1, D^2, \dots, D^T) \in \mathbb{R}_+^{TJ \times TJ}, \quad (44)$$

where D^t ($t = 1, \dots, T$) denotes a diagonal matrix with its diagonal entries

$$d_{jj}^t = \begin{cases} \frac{[A^T A X + \alpha X X^T X + \Lambda Y]_{jt} + \delta_{\mathbf{X}}^k}{\bar{x}_{jt}^k}, & \text{if } j \in \mathbf{I}_n, \\ 0, & \text{otherwise.} \end{cases} \quad (45)$$

$$\nabla_{\mathbf{X}}^2 L(\mathbf{X}^{(k)}) = \begin{pmatrix} \nabla_{\mathbf{X}}^2 L(\mathbf{X}^{(k)})_1 & & \\ & \nabla_{\mathbf{X}}^2 L(\mathbf{X}^{(k)})_2 & \\ & & \ddots \\ & & & \nabla_{\mathbf{X}}^2 L(\mathbf{X}^{(k)})_T \end{pmatrix} \in \mathbb{R}_+^{TJ \times TJ}, \quad (47)$$

where $\nabla_{\mathbf{X}}^2 L(\mathbf{X}^{(k)})_n$ denotes the n th column of $\nabla_{\mathbf{X}}^2 L(\mathbf{X}^{(k)})$. Next, for the function $G(\mathbf{X}, \mathbf{X}^{(k)})$, which is an auxiliary of $L(\mathbf{X})$, we first have to prove the following Lemma 2. As shown later, $A^{(k)}$ and $X^{(k)}$ must be bounded for $L(A^{(k)}, X^{(k)})$. The boundedness of $A^{(k)}$ and $X^{(k)}$ will be proven in Theorem 3.

Lemma 2. *Given sufficiently large $\delta_{\mathbf{X}}^{(k)}$ and the boundedness of $A^{(k)}$ and, then*

- (1) $G(\mathbf{X}, \mathbf{X}) = L(\mathbf{X})$.
- (2) $G(\mathbf{X}^{(k)}, \mathbf{X}^{(k)}) = L(\mathbf{X}^{(k)})$.
- (3) $G(\mathbf{X}, \mathbf{X}) \leq G(\mathbf{X}, \mathbf{X}^{(k)})$. Moreover, $G(\mathbf{X}, \mathbf{X}) = G(\mathbf{X}, \mathbf{X}^{(k)})$ only if $X^{(k)}$ satisfies the KKT conditions.
- (4) $G(\mathbf{X}, \mathbf{X}^{(k)}) \leq G(\mathbf{X}^{(k)}, \mathbf{X}^{(k)})$, and the equality holds only if $X^{(k)}$ satisfies the KKT conditions.

Proof. (1) and (2) are obvious from equation (41).

From (3), because $G(\mathbf{X}, \mathbf{X}) = L(\mathbf{X})$, we have

$$G(\mathbf{X}, \mathbf{X}^{(k)}) - L(\mathbf{X}) = \frac{1}{2} \sum_{t=1}^T \left\{ \sum_{j=1}^J \hat{x}_{jt}^2 \bar{d}_{jj}^t + \delta_{\mathbf{X}}^{(k)} \sum_{j=1}^J \hat{x}_{jt}^2 \hat{d}_{jj}^t \right\} - \frac{1}{2} \sum_{t=1}^T \left[(\hat{x}_t^{(k)})^T L^t \hat{x}_t \right] - \xi_{\mathbf{X}}^{(k)}. \quad (51)$$

With the boundedness of $A^{(k)}$ and $X^{(k)}$ and using sufficiently large $\delta_{\mathbf{X}}^{(k)}$, the inequality $G(\mathbf{X}, \mathbf{X}) \leq G(\mathbf{X}, \mathbf{X}^{(k)})$ can be guaranteed.

Additionally, it is clear that $G(\mathbf{X}, \mathbf{X}) = G(\mathbf{X}, \mathbf{X}^{(k)})$ only if $X^{(k)}$ satisfies the KKT conditions demonstrated in equation

Here,

$\mathbf{I}_n = \{j | x_{jt}^k > 0, \nabla_{\mathbf{X}} L(\mathbf{X}^{(k)})_{jt} \neq 0, \text{ or } x_{jt}^k = 0, \nabla_{\mathbf{X}} L(\mathbf{X}^{(k)})_{jt} < 0\}$ denotes the set of non-KKT indices in the n th column of $\mathbf{X}^{(k)}$.

Using the Taylor series, the expansion formulation for $L(\mathbf{X})$ can be expressed as follows:

$$L(\mathbf{X}) = L(\mathbf{X}^{(k)}) + \text{tr} \left\{ (\mathbf{X} - \mathbf{X}^{(k)})^T \nabla_{\mathbf{X}} L(\mathbf{X}^{(k)}) \right\} + \frac{1}{2} \text{tr} \left\{ (\mathbf{X} - \mathbf{X}^{(k)})^T \mathbf{L} (\mathbf{X} - \mathbf{X}^{(k)}) \right\} + \xi_{\mathbf{X}}^{(k)}, \quad (46)$$

where $\mathbf{L} = \nabla_{\mathbf{X}}^2 L(\mathbf{X}^{(k)})$ and $\xi_{\mathbf{X}}^{(k)}$ represent the higher components of the Taylor series, and

$$G(\mathbf{X}, \mathbf{X}^{(k)}) - G(\mathbf{X}, \mathbf{X}) = G(\mathbf{X}, \mathbf{X}) - L(\mathbf{X})$$

$$= \frac{1}{2} \text{tr} \left\{ (\mathbf{X} - \mathbf{X}^{(k)})^T (\mathbf{D} - \mathbf{L}) (\mathbf{X} - \mathbf{X}^{(k)}) \right\} - \xi_{\mathbf{X}}^{(k)} \quad (48)$$

$$= \sum_{t=1}^T \left[(\mathbf{x}_t - \mathbf{x}_t^{(k)})^T (D^t - L^t) (\mathbf{x}_t - \mathbf{x}_t^{(k)}) \right] - \xi_{\mathbf{X}}^{(k)},$$

$$D^t - L^t = \bar{D}^t + \delta_{\mathbf{X}}^{(k)} \hat{D}^t - L^t, \quad (49)$$

where $\bar{D}^t$ and $\delta_{\mathbf{X}}^{(k)} \hat{D}^t$ are diagonal matrices that sum up to D^t with

$$\bar{d}_{jj}^t = \begin{cases} \frac{[\nabla_{\mathbf{X}} D_F(\mathbf{X})]_{jt}^+}{\bar{x}_{jt}^k}, & \text{if } j \in \mathbf{I}_n, \\ 0, & \text{otherwise,} \end{cases}$$

$$\hat{d}_{jj}^t = \begin{cases} \frac{1}{\bar{x}_{jt}^k}, & \text{if } j \in \mathbf{I}_n, \\ 0, & \text{otherwise.} \end{cases} \quad (50)$$

Let $\mathbf{x}_t - \mathbf{x}_t^{(k)} = \hat{\mathbf{x}}_t$; then,

(48) regardless of $\xi_{\mathbf{X}}^{(k)}$. Because $\delta_{\mathbf{X}}^{(k)}$ is a variable, $G(\mathbf{X}, \mathbf{X}) = G(\mathbf{X}, \mathbf{X}^{(k)})$ holds only if $X = X^{(k)}$ in equation (51) as a result of the update rule of X and the boundedness of $A^{(k)}$ and $X^{(k)}$.

(4) Using

$$G(\mathbf{X}^{(k)}, \mathbf{X}^{(k)}) - G(\mathbf{X}, \mathbf{X}^{(k)}) = -tr\left\{(\mathbf{X} - \mathbf{X}^{(k)})^T \nabla_{\mathbf{X}} L(\mathbf{X}^{(k)}) (\mathbf{X} - \mathbf{X}^{(k)})\right\} \\ - \frac{1}{2} tr\left\{(\mathbf{X} - \mathbf{X}^{(k)})^T \mathbf{D}(\mathbf{X} - \mathbf{X}^{(k)})\right\}, \quad (52)$$

and the positive semidefinite of matrix $\mathbf{D}$, $G(\mathbf{X}, \mathbf{X}^{(k)})$ is a strictly convex function with respect to $\mathbf{X}^{(k)}$; it has a unique minimum that satisfies

$$\mathbf{D}(\mathbf{X} - \mathbf{X}^{(k)}) + \nabla_{\mathbf{X}} L(\mathbf{X}^{(k)}) = 0. \quad (53)$$

The aforementioned consideration results in the following:

$$G(\mathbf{X}^{(k)}, \mathbf{X}^{(k)}) - G(\mathbf{X}, \mathbf{X}^{(k)}) = \frac{1}{2} tr\left\{(\mathbf{X} - \mathbf{X}^{(k)})^T \mathbf{D}(\mathbf{X} - \mathbf{X}^{(k)})\right\} \geq 0. \quad (54)$$

If $\mathbf{X}^{(k)}$ satisfies the KKT conditions, then $\mathbf{X} = \mathbf{X}^{(k)}$, and therefore, $G(\mathbf{X}^{(k)}, \mathbf{X}^{(k)}) = G(\mathbf{X}, \mathbf{X}^{(k)})$.

Conversely, assuming that the equality holds, but $\mathbf{X}^{(k)}$ does not satisfy the KKT conditions, there is at least an index such that

$$x_{jt} \neq \bar{x}_{jt}^{(k)}, \\ \frac{[\nabla_{\mathbf{X}} D_F(\mathbf{X})]_{jt}^+ + \delta_X^k}{\bar{x}_{jt}^k} \geq \frac{\delta_X^k}{\bar{x}_{jt}^k}. \quad (55)$$

Furthermore, if $\bar{x}_{jt}^{(k)} = 0$, then $x_{jt} = x_{jt}^{(k)}$, which contradicts the equality assumption. Therefore, $\bar{x}_{jt}^{(k)} \neq 0$. As a result,

$$G(\mathbf{X}^{(k)}, \mathbf{X}^{(k)}) - G(\mathbf{X}, \mathbf{X}^{(k)}) \geq \frac{(x_{jt} - x_{jt}^{(k)}) \delta_X^k}{\bar{x}_{jt}^{(k)}} > 0, \quad (56)$$

which contradicts the assumption. $\square$

Theorem 1. *Given a sufficiently large number $\delta_X^{(k)}$ and supposing that $A^{(k)}$ and $X^{(k)}$ are bounded, then for the auxiliary function G ,*

- (1) $L(X^{(k+1)}) \leq L(X^{(k)})$ under the update rule in equation (4) and the equality holds only if $X^{(k)}$ satisfies the KKT conditions
- (2) $L(A^{(k+1)}) \leq L(A^{(k)})$ under the update rule in equation (3) and the equality holds only if $A^{(k)}$ satisfies the KKT conditions
- (3) $L(A^{(k+1)}, X^{(k+1)}) \leq L(A^{(k)}, X^{(k)})$ and the equality holds only if $(A^{(k)}, X^{(k)})$ is a stationary point

Proof. (1) and (2) can be obtained directly from Lemma 1. (3) By combining (1) and (2), it is obvious that

$$L(A^{(k+1)}, X^{(k+1)}) \leq L(A^{(k)}, X^{(k)}), \quad (57)$$

with $L(A^{(k+1)}, X^{(k+1)}) = L(A^{(k)}, X^{(k)})$ only if $A^{(k)}$ and $X^{(k)}$ satisfy the KKT conditions, i.e., $(A^{(k)}, X^{(k)})$ is a stationary point. $\square$

Theorem 2. *The objective function (22) is nonincreasing under the multiplicative update rules in equations (25) and (26).*

Proof. This theorem is the corollary of Theorem 1. $\square$

Lemma 3. *Sequence $\{A^{(k)}, X^{(k)}, \Lambda^{(k)}\}$ generated by Algorithm 2 is bounded and $L(A^{(k)}, X^{(k)})$ is lower bounded.*

Proof. Let us verify Assumptions A1–A3 (assumptions A1–A3 can be found in the works of [23]). Assumption A1 holds because of the coercivity of $\|\cdot\|_F$. Assumptions A2 and A3 hold because all the coefficient matrices are identity matrices. Additionally, because $\|\cdot\|_F$ is Lipschitz differentiable, $\{A^{(k)}, X^{(k)}, \Lambda^{(k)}\}$ generated by Algorithm 3 is bounded, and $L(A^{(k)}, X^{(k)})$ is lower bounded according to Theorem 2 presented in the works of [23].

The convergence property of the modified update rules in equations (27) and (29) is stated as follows: $\square$

Theorem 3. *Given a sufficiently large number $\delta_X^{(k)}$, every limit point of this algorithm is a stationary point of the following optimization problem:*

$$\min_{A \geq \varepsilon, X \geq \varepsilon} L(A, X) = \frac{1}{2} \|Y - AX\|_F^2 \\ + \frac{\alpha}{2} \|XX^T - I\|_F^2 + \Lambda \cdot (XX^T - I). \quad (58)$$

Proof. Because $\{A^{(k)}, X^{(k)}\} (k = 1, \dots, \infty)$ are in a compact set, the sequence $\{A^{(k)}, X^{(k)}\}$ generated by Algorithm 2 has at least one limit point. Gillis and Glineur [23] introduced a similar analysis that every limit point of this algorithm is a stationary point of the optimization problem (41). $\square$

3.3. Auxiliary Variable-Based ADM Extension to ONMF. To facilitate the efficient use of ADM, we first introduce two auxiliary variables, U and V , and consider the following equivalent model:

$$\min_{X, Y, U, V} \frac{1}{2} \|Y - AX\|_F^2 \quad (59)$$

$$\text{s.t. } A = U, X = V, U \geq 0, V \geq 0, XX^T = I,$$

where $A, U \in \mathbb{R}^{m \times q}$ and $Y, V \in \mathbb{R}^{q \times n}$. The augmented Lagrangian of equation (59) is as follows:

$$\begin{aligned} \mathcal{L}_A(A, X, U, V, \Lambda, \Pi) = & \frac{1}{2} \|Y - AX\|_F^2 + \frac{\alpha}{2} \|XX^T - I\|_F^2 + \Lambda \cdot (A - U) \\ & + \Pi \cdot (X - V) + \frac{\rho_1}{2} \|A - U\|_F^2 + \frac{\rho_2}{2} \|X - V\|_F^2, \end{aligned} \quad (60)$$

where $\Lambda \in \mathbb{R}^{m \times q}$, $\Pi \in \mathbb{R}^{q \times n}$ are Lagrangian multipliers, $\alpha, \rho_1, \rho_2 > 0$ are penalty parameters, and $A \cdot B := \sum_{i,j} a_{ij} b_{ij}$ for matrices A and B of the same size.

The alternating direction method for equation (59) is derived by successively minimizing $\mathcal{L}_A$ with respect to A, X, U, V , one at a time, while fixing others at their more recent values, i. e.,

$$\begin{aligned} A_{k+1} &= \arg \min \mathcal{L}_A(A, X_k, U_k, V_k, \Lambda_k, \Pi_k), \\ X_{k+1} &= \arg \min \mathcal{L}_A(A_{k+1}, X, U_k, V_k, \Lambda_k, \Pi_k), \\ U_{k+1} &= \arg \min_{U \geq 0} \mathcal{L}_A(A_{k+1}, X_{k+1}, U, V_k, \Lambda_k, \Pi_k), \\ V_{k+1} &= \arg \min_{V \geq 0} \mathcal{L}_A(A_{k+1}, X_{k+1}, U_{k+1}, V, \Lambda_k, \Pi_k), \end{aligned} \quad (61)$$

and then updating the multipliers Λ and Π . The introduction of the two auxiliary variables, U and V , makes it easy to carry out each of the alternating minimization steps. Specifically, these steps can be expressed in closed form as follows:

$$a_{ij} \leftarrow a_{ij} \left(\frac{[YX^T + \rho_1 U]_{ij}}{[AXX^T + \Pi + \rho_1 A]_{ij} + \delta} \right)^\omega, \quad (62)$$

$$x_{jt} \leftarrow x_{jt} \left(\frac{[A^T Y + \alpha X + \rho_2 V]_{jt}}{[A^T A X + \alpha X X^T X + \Lambda + \rho_2 X]_{jt} + \delta} \right)^\omega, \quad (63)$$

$$U \leftarrow \mathcal{P}_+ \left(A + \frac{\Lambda}{\rho_1} \right), \quad (64)$$

$$V \leftarrow \mathcal{P}_+ \left(X + \frac{\Pi}{\rho_2} \right), \quad (65)$$

$$\Lambda \leftarrow \Lambda + \gamma \rho_1 (A - U), \quad (66)$$

$$\Pi \leftarrow \Pi + \gamma \rho_2 (X - V). \quad (67)$$

The detailed procedure for the algorithm of Problem 59 can be seen in Algorithm 3.

4. Experiments and Results

In this section, we analyze and compare Algorithms 2 and 3 with Algorithms 1 and 4 (ONP-MF [18]). All experiments are performed using the Windows 7 operating system and MATLAB v8.1 (R2013a) running on a Lenovo laptop with an Intel Core(TM)i5-3470 CPU at 3.2 GHz and 4 GB of memory.

4.1. Test on Images. To visualize the results, we use examples of ONMF arising in image processing to analyze the quality of the solutions. Two grayscale images are tested using

Algorithms 1–3 and 4. The two grayscale images, a panda in Figure 1(a) and a cat in Figure 1(b), have resolutions of 1200×1920 and 128×128 , respectively.

We evaluate the degree of approximation and orthogonality using two indices:

$$\text{Relative residual error (RRE): } \frac{\|Y - AX\|_F}{\|Y\|_F}, \quad (68)$$

$$\text{Orthogonality residual error (ORE): } \|XX^T - I\|_F.$$

We set $\text{tol} = 10^{-5}$, $\delta = 10^{-8}$, and $\alpha = 0.1$, and $\text{maxiter} = 500$ for the image of the cat and $\text{maxiter} = 100$ for the image of the panda, after which we repeat the process 10 times and calculate the relative errors and orthogonality. The results are given in Tables 1–5, and the recovered images are shown in Figures 2 and 3.

Tables 1 and 2 show that Algorithm 2 performs slightly better than Algorithms 1 and 3 in terms of relative errors in most cases. Moreover, we observe that Algorithm 4 converges in least iterations. From Tables 3 and 4, we can see that Algorithm 2 demonstrates the best performance compared with Algorithms 1, 3 and 4 in terms of orthogonality. From Table 5, we observe that Algorithm 4 requires the lowest computational cost compared with the rest of algorithms. From Figures 2 and 3, we observe that Algorithm 2 achieves a better recovery quality of images compared with Algorithms 1 and 3, and the recovery quality of Algorithm 4 is the best.

4.2. Clustering Capability. We test the clustering capability of the three ONMF algorithms on the ALLAML gene-sample data provided by [4] and compare the performance of the algorithms. To measure the clustering performance, we use purity and entropy as our performance metrics. We expect that these metrics will provide us with good insights on how our algorithm works.

Entropy measures how classes of genes are distributed within each cluster [25]. It is defined as follows:

$$\text{entropy} \equiv \frac{1}{N \log_2 S} \sum_{r=1}^R \sum_{s=1}^S c_{rs} \log_2 \left(\frac{c_{rs}}{c_r} \right), \quad (69)$$

where N denotes the number of samples, c_{rs} denotes the number of samples in the r th cluster that belong to the s th class, and c_r denotes the size of the r th cluster.

Purity is the most commonly used metric. It measures the extent to which each cluster contains data points from primarily one class [25]. Therefore, the higher the value of purity, the better the clustering performance. Purity is defined as follows:

$$\text{purity} \equiv \frac{1}{N} \sum_{r=1}^R \max_s c_{rs}. \quad (70)$$

FIGURE 1: Original images. (a) Panda. (b) Cat.

TABLE 1: Comparison of **REE** testing on image panda.

Rank J	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
10	0.2256 ± 0.0216	0.2096 ± 0.0276	0.2413 ± 0.0355	0.2191 ± 0.0227
20	0.1973 ± 0.0173	0.1677 ± 0.0183	0.1811 ± 0.0179	0.1761 ± 0.0166
40	0.1876 ± 0.0125	0.1310 ± 0.0156	0.1665 ± 0.0142	0.1296 ± 0.0114
60	0.1722 ± 0.0057	0.1278 ± 0.0047	0.1542 ± 0.0113	0.1204 ± 0.0086
80	0.1580 ± 0.0063	0.1105 ± 0.0081	0.1401 ± 0.0086	0.1079 ± 0.0068

TABLE 2: Comparison of **REE** testing on image cat.

Rank J	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
5	0.1906 ± 0.0314	0.1855 ± 0.0191	0.1957 ± 0.0241	0.1947 ± 0.0264
10	0.1761 ± 0.0245	0.1638 ± 0.0176	0.1740 ± 0.0211	0.1711 ± 0.0216
15	0.1659 ± 0.0174	0.1513 ± 0.0152	0.1601 ± 0.0187	0.1473 ± 0.0182
20	0.1584 ± 0.0112	0.1362 ± 0.0121	0.1474 ± 0.0158	0.1384 ± 0.0155
25	0.1521 ± 0.0121	0.1252 ± 0.0102	0.1392 ± 0.0129	0.1234 ± 0.0133
30	0.1477 ± 0.0189	0.1175 ± 0.0093	0.1292 ± 0.0111	0.1103 ± 0.0115

TABLE 3: Comparison of **ORE** testing on image cat.

Rank J	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
5	1.1121 ± 0.3217	0.4588 ± 0.2211	0.9992 ± 0.0001	0.9091 ± 0.3015
10	2.7271 ± 0.2473	0.5600 ± 0.2008	0.9994 ± 0.0002	0.9524 ± 0.2182
15	4.3523 ± 0.1820	0.5681 ± 0.1656	0.9996 ± 0.0001	0.9677 ± 0.1796
20	5.8419 ± 0.1556	0.5727 ± 0.1357	0.9996 ± 0.0003	0.9756 ± 0.1562
25	7.8002 ± 0.1482	0.5799 ± 0.1229	0.9997 ± 0.0002	0.9804 ± 0.1400
30	9.4566 ± 0.1331	0.5860 ± 0.0980	0.9998 ± 0.0002	0.9836 ± 0.1280

TABLE 4: Comparison of **ORE** testing on image panda.

Rank J	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
10	372.6551 ± 10.2546	0.5457 ± 0.1755	0.9955 ± 0.1257	0.9210 ± 0.1546
20	686.1020 ± 40.1120	0.5601 ± 0.1815	0.9979 ± 0.1522	0.9664 ± 0.1753
40	$1.3621 \times 10^3 \pm 70.1885$	0.6573 ± 0.1800	0.9995 ± 0.2561	0.9718 ± 0.2051
60	$1.9141 \times 10^3 \pm 114.2850$	0.6607 ± 0.2111	0.9999 ± 0.2101	0.9957 ± 0.2689
80	$2.6000 \times 10^3 \pm 160.4782$	0.8610 ± 0.1566	1.0000 ± 0.2008	0.9980 ± 0.1280

TABLE 5: Comparison of cpu time (seconds) testing on image panda.

Rank J	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
10	93.1131 ± 5.2421	427.2460 ± 10.8541	87.0101 ± 6.0105	15.7797 ± 2.3085
20	96.2495 ± 4.2680	470.1573 ± 15.9630	98.3510 ± 8.1507	20.9627 ± 3.0343
40	99.1576 ± 8.0485	526.0207 ± 18.0012	105.1883 ± 9.2057	36.4698 ± 5.2240
60	101.6872 ± 10.5701	579.2561 ± 23.2007	112.1097 ± 10.0005	58.2959 ± 8.4178
80	104.0500 ± 9.4910	819.0018 ± 30.0804	120.2542 ± 13.5076	90.9566 ± 13.0304

FIGURE 2: Recovered 128×128 cat by Algorithms 1–3, and Algorithm 4. (a) $J=5$. (b) $J=5$. (c) $J=5$. (d) $J=5$. (e) $J=15$. (f) $J=15$. (g) $J=15$. (h) $J=15$. (i) $J=30$. (j) $J=30$. (k) $J=30$. (l) $J=30$. FIGURE 3: Recovered 1200×1920 panda by Algorithms 1–3, and Algorithm 4. (a) $J=10$. (b) $J=10$. (c) $J=10$. (d) $J=10$. (e) $J=20$. (f) $J=20$. (g) $J=20$. (h) $J=20$. (i) $J=40$. (j) $J=40$. (k) $J=40$. (l) $J=40$. (m) $J=80$. (n) $J=80$. (o) $J=80$. (p) $J=80$.

TABLE 6: Performance $k=2$ on the leukemia data matrix.

	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
Purity	0.9216 ± 0.1256	0.9501 ± 0.0761	0.9266 ± 0.1127	0.6708 ± 0.0950
Entropy	0.3053 ± 0.1473	0.2390 ± 0.0655	0.3317 ± 0.0952	0.6653 ± 0.0958

TABLE 7: Performance $k=3$ on the leukemia data matrix.

	Algorithm 1	Algorithm 2	Algorithm 3	Algorithm 4
Purity	0.8912 ± 0.1024	0.9360 ± 0.1586	0.9218 ± 0.1895	0.8256 ± 0.1179
Entropy	0.2500 ± 0.0567	0.1796 ± 0.0465	0.2054 ± 0.0723	0.4132 ± 0.0590

Algorithms 2 and 3 are compared with Algorithms 1 and 4, and the comparisons are shown in Tables 6 and 7. From Tables 6 and 7, we observe that our Algorithms 2 and 3 clustering achieves better purity and entropy results on the leukemia data sets compared with the performance of Algorithms 1 and 4. There exist slight differences in the relative performance of purity and entropy in our comparison, that is, higher purity values do not necessarily correspond to lower entropy values. This is because entropy takes into account the entire distribution of the genes in a particular cluster and not only the largest class as in the computation of purity. In summary, the comparison shows that Algorithms 2 and 3 are viable and competitive in clustering tasks.

5. Conclusion

In this study, we develop two efficient modified algorithms based on the ADM algorithms for the ONMF problem. We then prove that any sequence of solutions generated by the modified updates of Algorithms 2 and 3 has at least one convergent subsequence, and the limit of any convergent subsequence is a stationary point of the corresponding optimization problem. From the numerical results, a notable observation is that our Algorithms 2 and 3 are feasible and efficient, especially in the properties of the solution, that is, the orthogonality.

The framework for the convergence analysis of the updates for Algorithm 3 presented in this article may be applied to the ADM algorithms for NMF [26].

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was partially supported by the Xijing University School Fund Project (no. XJ200102), the NSF of Shannxi Province (no. 20JK0963), and the NSF of China under Grant (no. 11171270).

References

- [1] 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, pp. 111–126, 2001.
- [2] D. Lee and H. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, 1999.
- [3] D. Lee and H. Seung, "Algorithms for non-negative matrix factorization," *Advances in Neural Processing Information Systems*, vol. 13, pp. 556–562, 2001.
- [4] J.-P. Brunet, P. Tamayo, T. R. Golub, and J. P. Mesirov, "Metagenes and molecular pattern discovery using matrix factorization," *Proceedings of the National Academy of Sciences*, vol. 101, no. 12, pp. 4164–4169, 2004.
- [5] M. Cooper and J. Foote, "Summarizing video using non-negative similarity matrix factorization," in *Proceedings of the IEEE Workshop on Multimedia Signal Processing*, pp. 25–28, St. Thomas, VI, USA, December 2002.
- [6] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *Journal of Machine Learning Research*, vol. 5, pp. 1457–1469, 2004.
- [7] S. Z. Li, X. Hou, H. Zhang, and Q. Cheng, "Learning spatially localized, parts-based representation," in *Proceedings of IEEE Computer Vision and Pattern Recognition*, pp. 207–212, Kauai, HI, USA, December 2001.
- [8] J.-X. Liu, D. Wang, Y.-L. Gao, C.-H. Zheng, Y. Xu, and J. Yu, "Regularized non-negative matrix factorization for identifying differentially expressed genes and clustering samples: a survey," *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, vol. 15, no. 3, pp. 974–987, 2018.
- [9] 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, 1994.
- [10] F. Sha, L. K. Saul, and D. D. Lee, "Multiplicative updates for nonnegative quadratic programming in support vector machines," *Advances in Neural Information Processing Systems*, vol. 15, pp. 1041–1048, 2003.
- [11] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, pp. 267–273, Toronto, Canada, July 2003.
- [12] N. Yu, M. J. Wu, J. X. Liu, C. H. Zheng, and Y. Xu, "Correntropy-based hypergraph regularized NMF for clustering and feature selection on multi-cancer integrated data," *IEEE Transactions on Cybernetics*, vol. 99, pp. 1–12, 2020.
- [13] R. Zdunek and A. Cichocki, "Non-negative matrix factorization with quasi-Newton optimization," in *Proceedings of*

- the Eighth International Conference on Artificial Intelligence and Soft Computing, ICAISC*, pp. 870–879, Springer, Zakopane, Poland, June 2006.
- [14] C. J. Lin, “Projected gradient methods for non-negative matrix factorization,” Technical report, Department of Computer Science, National Taiwan University, Taipei, Taiwan, 2005.
 - [15] S. Choi, “Algorithms for orthogonal nonnegative matrix factorization,” in *Proceedings of the IEEE International Joint Conference on Neural Networks, 2008, IJCNN 2008*, pp. 1828–1832, IEEE World Congress on Computational Intelligence, Hong Kong, China, June 2008.
 - [16] C. Ding, T. Li, W. Peng, and H. Park, “Orthogonal non-negative matrix tri-factorizations for clustering,” in *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Ser. KDD’06*, pp. 126–135, Philadelphia, PA, USA, August 2006.
 - [17] B. Bo Li, G. Guoxu Zhou, and A. Cichocki, “Two efficient algorithms for approximately orthogonal nonnegative matrix factorization,” *IEEE Signal Processing Letters*, vol. 22, no. 7, pp. 843–846, 2015.
 - [18] F. Pompili, N. Gillis, P. A. Absil, and F. Glineur, “Two algorithms for orthogonal nonnegative matrix factorization with application to clustering,” *Neurocomputing*, vol. 141, pp. 15–25, 2014.
 - [19] Z. Zhirong Yang and E. Oja, “Linear and nonlinear projective nonnegative matrix factorization,” *IEEE Transactions on Neural Networks*, vol. 21, no. 5, pp. 734–749, 2010.
 - [20] C. J. Lin, “On the convergence of multiplicative update algorithms for nonnegative matrix factorization,” *IEEE Transactions on Neural Networks*, vol. 18, no. 6, pp. 1589–1596, 2007.
 - [21] N. Gillis and F. Glineur, “Nonnegative factorization and the maximum edge Biclique problem,” 2008.
 - [22] A. Mirzal, “A convergent algorithm for orthogonal non-negative matrix factorization,” *Journal of Computational and Applied Mathematics*, vol. 260, pp. 149–166, 2014.
 - [23] N. Gillis and F. Glineur, “Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization,” *Neural Computation*, vol. 24, no. 4, pp. 1085–1105, 2012.
 - [24] J.-H. Yoo and S.-J. Choi, “Nonnegative matrix factorization with orthogonality constraints,” *Journal of Computing Science and Engineering*, vol. 4, no. 2, pp. 97–109, 2010.
 - [25] Y. Zhao and G. Karypis, “Empirical and theoretical comparisons of selected criterion functions for document clustering,” *Machine Learning*, vol. 55, no. 3, pp. 311–331, 2004.
 - [26] Y. Zhang, “An alternating direction algorithm for nonnegative matrix factorization,” Rich technical report, Rice University, Houston, TX, USA, 2010.