A convergent algorithm for bi-orthogonal nonnegative matrix tri-factorization

Andri Mirzal

Received: 23 October 2019 / Revised: 4 April 2021 / Accepted: 3 June 2021 /
Published online: 8 October 2021
© Springer-Verlag GmbH Germany, part of Springer Nature 2021

Abstract

A convergent algorithm for nonnegative matrix factorization (NMF) with orthogonality constraints imposed on both basis and coefficient matrices is proposed in this paper. This factorization concept was first introduced by Ding et al. (Proceedings of 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 126–135, 2006) with intent to further improve clustering capability of NMF. However, as the original algorithm was developed based on multiplicative update rules, the convergence of the algorithm cannot be guaranteed. In this paper, we utilize the technique presented in our previous work Mirzal (J Comput Appl Math 260:149–166, 2014a; Proceedings of the first international conference on advanced data and information engineering (DaEng-2013). Springer, pp 177–184, 2014b; IEEE/ACM Trans Comput Biol Bioinform 11(6):1208–1217, 2014c) to develop a convergent algorithm for this problem and prove that it converges to a stationary point inside the solution space. As it is very hard to numerically show the convergence of an NMF algorithm due to the slow convergence and numerical precision issues, experiments are instead performed to evaluate whether the algorithm has the nonincreasing property (a necessary condition for the convergence) where it is shown that the algorithm has this property. Further, clustering capability of the algorithm is also inspected by using Reuters-21578 data corpus.

Keywords Clustering methods · Convergent algorithm · Nonnegative matrix factorization · Orthogonality constraint

Mathematics Subject Classification 65F30 · 15A23

Andri Mirzal
andri.mirzal@kfupm.edu.sa

1 Information and Computer Sciences Department, King Fahd University of Petroleum and Minerals, Dhahran, Kingdom of Saudi Arabia
1 Introduction

Nonnegative matrix factorization (NMF) was initially introduced with an intention to produce a meaningful decomposition for learning parts of faces from facial images and semantic features of articles (Lee and Seung 1999). In the basic form, NMF seeks to decompose a nonnegative matrix into a pair of other nonnegative matrices with lower ranks:

\[ A \approx BC, \]

where \( A \in \mathbb{R}^{M \times N} = [a_1, \ldots, a_N] \) denotes the data matrix, \( B \in \mathbb{R}^{M \times R} = [b_1, \ldots, b_R] \) denotes the basis matrix, \( C \in \mathbb{R}^{R \times N} = [c_1, \ldots, c_N] \) denotes the coefficient matrix, \( R \) denotes the number of factors which usually is chosen so that \( R \ll \min(M, N) \), and \( \mathbb{R}^{M \times N}_+ \) denotes \( M \) by \( N \) nonnegative real matrix. The following minimization problem is usually solved to compute \( B \) and \( C \):

\[
\min_{B,C} J(B, C) = \frac{1}{2} \| A - BC \|^2_F \text{ s.t. } B \succeq 0, C \succeq 0, \tag{1}
\]

where \( \| \cdot \|_F^2 \) denotes the Frobenius norm, and the symbol \( \succeq \) denotes entrywise bigger-than-or-equal comparison.

In (1999), Lee and Seung proposed a multiplicative update rules (MUR) based algorithm to solve the minimization problem in Eq. 1, and showed that the algorithm is monotonically nonincreasing (Lee and Seung 2000). Following this breakthrough, many works rapidly proposed other MUR based NMF algorithms for solving slightly different objective functions. Due to a minor issue in Lee and Seung (2000), where the authors incorrectly considered monotonic nonincreasing property as convergence guarantee, many works that followed also considered their MUR based NMF algorithms have convergence guarantee even though the provided proofs are for the nonincreasing property. This issue has been made clear by Lin in 2007, where he stated that proving the convergence of a MUR based NMF algorithm is difficult and instead introduced a clever trick to modify the Lee and Seung’s algorithm into an additive update rules (AUR) based algorithm and proved its convergence to a stationary point inside the solution space. It is worth to mention that even though MUR based NMF algorithm by Lee and Seung has the nonincreasing property, it does not necessarily mean that any MUR based NMF algorithm also has the nonincreasing property. For example MUR based algorithm for Tikhonov regularized NMF may lose this property when the regularization parameters are too large (Berry et al. 2006; Mirzal 2014c).

In a theoretical work by Zhao and Tan (2017), the authors proposed a unified framework for MUR based NMF algorithms with convergence guarantee. This framework uses the general \( h \)-divergence which subsumes many distance measures including Frobenius norm, Kullback-Leibler, Itakura-Saito, Hellinger, \( \alpha (\alpha \neq 0), \beta, \gamma, \alpha - \beta \), and Renyi divergences, and includes the two most discussed constraints in NMF, i.e., sparseness constraints (by using \( L_{1,1} \) regularization) and smoothness constraints (by using Tikhonov regularization). The following gives the NMF objective function of the framework:
\[
\min_{B, C} J(B, C) = D(A \| BC) + \frac{1}{2} \beta_1 \| B \|_F^2 + \beta_2 \| B \|_{1,1} + \frac{1}{2} \alpha_1 \| C \|_F^2 + \alpha_2 \| C \|_{1,1}
\]
\[\text{s.t. } B \succeq 0, C \succeq 0,\]

where \( D \) denotes the \( h \)-divergence; and \( \alpha_1 \geq 0, \beta_1 \geq 0, \alpha_2 > 0, \) and \( \beta_2 > 0 \) denote the regularization parameters. However, the convergence proof requires that sparseness constraints must be present to guarantee coarseness of the solutions, and no other constraints like orthogonality can be added. Thus this framework cannot be utilized to derive convergent algorithms for many other cases including the standard NMF (Eq. 1) and orthogonal NMFs.

A convergent algorithm based on additive update rules (AUR) for uni-orthogonal NMF (UNMF) has been presented in our previous work (Mirzal 2014a). In this paper, we utilized the technique developed in Mirzal (2014a) to design a convergent algorithm for bi-orthogonal nonnegative matrix tri-factorization (BNMtF). As orthogonality constraint cannot be recast into alternating nonnegativity-constrained least square (ANLS) framework (see Kim and Park 2007, 2008a for discussion on ANLS), some convergent algorithms for the standard NMF, e.g., (Kim and Park 2008a; Lin 2007; Kim et al. 2008, 2007; Kim and Park 2008; Lin 2005) cannot be extended to the problem. Moreover unlike the standard NMF or UNMF, BNMtF is a three-block alternating least squares (ALS) problem, and as shown in Grippo and Sciandrone (2000), Henseler (2010), an algorithm that converges in a two-block ALS problem not necessarily can be modified to converge in a three-block (or more) ALS problem. Thus, developing a convergent algorithm for BNMtF is not a trivial task.

2 Bi-orthogonal nonnegative matrix tri-factorization

In order to improve clustering capability of NMF, Ding et al. (2006) introduced two types of orthogonal NMFs: uni-orthogonal NMF (UNMF) and bi-orthogonal nonnegative matrix tri-factorization (BNMtF) where the former imposes orthogonality constraint on either columns of \( B \) or rows of \( C \), and the latter imposes orthogonality constraints on both columns of \( B \) and rows of \( C \) simultaneously. And due to the tight constraints in the latter, they introduced a third factor to absorb the approximation error. They then proposed a MUR based algorithm for each orthogonal NMF. The following describes the original BNMtF objective function proposed in Ding et al. (2006).

\[
\min_{B, C, S} J(B, C, S) = \frac{1}{2} \| A - BSC \|_F^2
\]
\[\text{s.t. } B \succeq 0, S \succeq 0, C \succeq 0, \frac{1}{2} (CC^T - I) = 0, \text{ and } \frac{1}{2} (B^TB - I) = 0 \quad (2)\]

where \( B \in \mathbb{R}_+^{M \times P} \) and \( C \in \mathbb{R}_+^{Q \times N} \); and \( S \in \mathbb{R}_+^{P \times Q} \) is the third factor introduced to absorb the scale differences of \( A \) and \( BC \) due to the strict orthogonality constraints on \( B \) and \( C \). We set \( P = Q \) for the rest of this paper to keep the same number of factors in \( B \) and \( C \) which is preferable for some applications such as co-clustering. Accordingly, the Karush-Kuhn-Tucker (KKT) function of the objective function can be defined as:
\[ L(B, C, S) = J(B, C, S) - \text{tr} \left( \Gamma_B B^T \right) - \text{tr} \left( \Gamma_S S^T \right) - \text{tr} \left( \Gamma_C C \right) \\
+ \frac{1}{2} \text{tr} \left( A_C (C C^T - I) \right) + \frac{1}{2} \text{tr} \left( A_B (B^T B - I) \right), \]

where \( \Gamma_B \in \mathbb{R}^{M \times P} \), \( \Gamma_S \in \mathbb{R}^{P \times Q} \), \( \Gamma_C \in \mathbb{R}^{N \times Q} \), \( A_C \in \mathbb{R}^{Q \times Q} \), and \( A_B \in \mathbb{R}^{P \times P} \) are the KKT multipliers.

Instead of solving the problem in Eq. 2, Ding et al. (2006) proposed to absorb the orthogonality constraints into the objective function.

\[
\min_{B, C, S} J(B, C, S) = \frac{1}{2} \| A - BSC \|_F^2 + \frac{1}{2} \text{tr} \left( A_C (C C^T - I) \right) \\
+ \frac{1}{2} \text{tr} \left( A_B (B^T B - I) \right). \tag{3}
\]

And accordingly, the KKT conditions for objective in Eq. 3 can be written as:

\[
\begin{align*}
B^* &\succeq 0, \\
S^* &\succeq 0, \\
C^* &\succeq 0, \\
\nabla_B J(B^*) &= \Gamma_B \succeq 0, \\
\nabla_S J(S^*) &= \Gamma_S \succeq 0, \\
\nabla_C J(C^*) &= \Gamma_C^\top \succeq 0, \\
\nabla_B J(B^*) \odot B^* &= 0, \\
\nabla_S J(S^*) \odot S^* &= 0, \\
\nabla_C J(C^*) \odot C^* &= 0,
\end{align*}
\]

where \( \odot \) denotes entrywise multiplication operation, and

\[
\begin{align*}
\nabla_B J(B) &= BSCC^T S^T - AC^T S^T + B3_B, \\
\nabla_C J(C) &= S^T B^T BSC - S^T B^T A + A_C C, \\
\nabla_S J(S) &= B^T BSCC^T - B^T AC^T.
\end{align*}
\]

Then, by using the multiplicative updates (Lee and Seung 2000), Ding et al. (2006) derived BNMtF algorithm as follows:

\[
\begin{align*}
b_{mp}^{\text{new}} &\leftarrow b_{mp} \frac{(AC^T S^T)_{mp}}{B(SCC^T S^T + A_B)_{mp}}, \\
c_{qn}^{\text{new}} &\leftarrow c_{qn} \frac{(S^T B^T A)_{qn}}{(S^T B^T BS + A_C)_{qn}}, \\
s_{pq}^{\text{new}} &\leftarrow s_{pq} \frac{(B^T AC^T)_{pq}}{(B^T BSCC^T)_{pq}},
\end{align*}
\tag{4-6}
\]

with

\[
A_B = B^T AC^T S^T - SCC^T S^T \quad \text{and} \quad A_C = S^T B^T AC^T - S^T B^T BS
\]

are derived exactly for the diagonal entries, and \textit{approximately} for off-diagonal entries by relaxing the nonnegativity constraints.
The complete BNMtF algorithm proposed in Ding et al. (2006) is shown in Algorithm 1 where \( \delta \) denotes some small positive number (note that the normalization step is not recommended as it will change the objective value). As there are approximations in deriving \( \mathbf{A}_B \) and \( \mathbf{A}_C \), Algorithm 1 may or may not be minimizing the objective Eq. 3. Further, the auxiliary function used by the authors to prove the nonincreasing property is for the algorithm in Eqs. 4 – 6, not for Algorithm 1. So there is no guarantee that Algorithm 1 has the nonincreasing property. Figure 1 shows error per iteration of Algorithm 1 in Reuters4 dataset (see Sect. 4 for detailed info about the dataset). As the Algorithm 1 not only does not have the nonincreasing property but also fails to minimize the objective function, it is clear that the assumptions used to obtain \( \mathbf{A}_B \) and \( \mathbf{A}_C \) are not acceptable.

Algorithm 1 Original BNMtF algorithm by Ding et al. (2006).

Initialize \( \mathbf{B}^{(0)}, \mathbf{C}^{(0)}, \) and \( \mathbf{S}^{(0)} \) with positive matrices to avoid zero locking.

for \( k = 0, \ldots, K \) do

\[
\begin{align*}
    b_{mp}^{(k+1)} &\leftarrow b_{mp}^{(k)} + \frac{(\mathbf{A}^{(k)T} \mathbf{S}^{(k)T})_{mp}}{(\mathbf{B}^{(k)T} \mathbf{B}^{(k)} + \mathbf{A}^{(k)T} \mathbf{S}^{(k)T})_{mp}} \quad \forall m, p \\
    c_{qn}^{(k+1)} &\leftarrow c_{qn}^{(k)} - \frac{(\mathbf{S}^{(k)T} \mathbf{B}^{(k+1)T})_{qn}}{(\mathbf{S}^{(k)T} \mathbf{B}^{(k+1)} + \mathbf{A}^{(k)T} \mathbf{C}^{(k)})_{qn}} \quad \forall q, n \\
    s_{pq}^{(k+1)} &\leftarrow s_{pq}^{(k)} + \frac{(\mathbf{B}^{(k+1)T} \mathbf{A}^{(k+1)T})_{pq}}{(\mathbf{B}^{(k+1)T} \mathbf{B}^{(k+1)} + \mathbf{S}^{(k)} \mathbf{C}^{(k+1)} \mathbf{C}^{(k+1)T})_{pq}} \quad \forall p, q
\end{align*}
\]

end for

3 A convergent algorithm for BNMtF

We define BNMtF problem with the following objective function:

\[
\min_{\mathbf{B}, \mathbf{C}, \mathbf{S}} J(\mathbf{B}, \mathbf{C}, \mathbf{S}) = \frac{1}{2} \| \mathbf{A} - \mathbf{BSC} \|^2_F + \frac{\alpha}{2} \| \mathbf{CC}^T - \mathbf{I} \|^2_F + \frac{\beta}{2} \| \mathbf{B}^T \mathbf{B} - \mathbf{I} \|^2_F
\]

s.t. \( \mathbf{B} \succeq 0, \mathbf{C} \succeq 0, \mathbf{S} \succeq 0, \) \( \mathbf{S} \succeq 0, \)

where \( \alpha \) and \( \beta \) are regularization parameters; and as the orthogonality constraints are absorbed into the objective function, the computed \( \mathbf{B} \) and \( \mathbf{C} \) will only be approximately orthogonal. Therefore, the KKT function of the objective can be written as:

\[
L(\mathbf{B}, \mathbf{C}, \mathbf{S}) = J(\mathbf{B}, \mathbf{C}, \mathbf{S}) - \text{tr} (\mathbf{\Gamma}_\mathbf{B} \mathbf{B}^T) - \text{tr} (\mathbf{\Gamma}_\mathbf{S} \mathbf{S}^T) - \text{tr} (\mathbf{\Gamma}_\mathbf{C} \mathbf{C}).
\]
And the KKT conditions are:

\[ \begin{align*}
    &B^* \succeq 0, \quad S^* \succeq 0, \quad C^* \succeq 0, \\
    &\nabla_B J(B^*) = \Gamma_B \succeq 0, \quad \nabla_S J(S^*) = \Gamma_S \succeq 0, \quad \nabla_C J(C^*) = \Gamma_C \succeq 0, \\
    &\nabla_B J(B^*) \odot B^* = 0, \quad \nabla_S J(S^*) \odot S^* = 0, \quad \nabla_C J(C^*) \odot C^* = 0.
\end{align*} \]

(8)

where

\[ \begin{align*}
    \nabla_B J(B) &= BSCC^T S^T - AC^T S^T + \beta BB^T B - \beta B, \\
    \nabla_C J(C) &= S^T B^T BSC - S^T B^T A + \alpha CC^T C - \alpha C, \\
    \nabla_S J(S) &= B^T BSCC^T - B^T AC^T.
\end{align*} \]

As shown by Lee and Seung (2000), a MUR based algorithm can be derived by utilizing complementary slackness in the KKT conditions (the last line in Eq. 8). Therefore, a MUR based algorithm for BNMtF problem in Eq. 7 can be defined as:

\[ \begin{align*}
    b_{mp} &\leftarrow b_{mp} \frac{(AC^T S^T + \beta B)_{mp}}{(BSCC^T S^T + \beta BB^T B)_{mp}}, \\
    c_{qn} &\leftarrow c_{qn} \frac{(S^T B^T A + \alpha C)_{qn}}{(S^T B^T BSC + \alpha CC^T C)_{qn}}, \\
    s_{pq} &\leftarrow s_{pq} \frac{(B^T AC^T)_{pq}}{(B^T BSCC^T)_{pq}}.
\end{align*} \]
The complete MUR algorithm is given in Algorithm 2, and the AUR version is given in Algorithm 3 (please see, e.g., (Lin 2007; Mirzal 2014a) for discussion about how to derive an AUR algorithm from the MUR counterpart). As shown, the AUR algorithm can be initialized using nonnegative matrices as it does not inherit the zero locking problem from its MUR counterpart.

**Algorithm 2** The MUR algorithm for BNMtF problem in Eq. 7 (MU-B).

Initialize $B^{(0)}$, $C^{(0)}$, and $S^{(0)}$ with positive matrices to avoid zero locking.

for $k = 0, \ldots, K$ do

\[
B^{(k+1)}_{mp} \leftarrow B^{(k)}_{mp} - \frac{1}{m_p} \left( (AC^{(k)})^T S^{(k)} + \beta B^{(k)} \right)_{mp} + \delta^{(k)}_B \quad \forall m, p
\]

\[
c^{(k+1)}_{qn} \leftarrow c^{(k)}_{qn} - \frac{1}{q_n} \left( (S^{(k)})^T A + \alpha C^{(k)} \right)_{qn} + \delta^{(k)}_C \quad \forall q, n
\]

\[
s^{(k+1)}_{pq} \leftarrow s^{(k)}_{pq} - \frac{1}{S_{pq}} \left( (B^{(k+1)})^T AC^{(k+1)} \right)_{pq} + \delta^{(k)}_S \quad \forall p, q
\]

end for

**Algorithm 3** The AUR algorithm for BNMtF problem in Eq. 7.

Initialize $B^{(0)}$, $C^{(0)}$, and $S^{(0)}$ with nonnegative matrices.

for $k = 0, \ldots, K$ do

\[
\bar{b}^{(k)}_{mp} \leftarrow B^{(k)}_{mp} - \frac{1}{m_p} \left( (AC^{(k)})^T S^{(k)} + \beta B^{(k)} \right)_{mp} + \delta^{(k)}_B \quad \forall m, p
\]

\[
\bar{c}^{(k)}_{qn} \leftarrow c^{(k)}_{qn} - \frac{1}{q_n} \left( (S^{(k)})^T A + \alpha C^{(k)} \right)_{qn} + \delta^{(k)}_C \quad \forall q, n
\]

\[
\bar{s}^{(k)}_{pq} \leftarrow s^{(k)}_{pq} - \frac{1}{S_{pq}} \left( (B^{(k+1)})^T AC^{(k+1)} \right)_{pq} + \delta^{(k)}_S \quad \forall p, q
\]

end for

There are the notations $\bar{b}^{(k)}_{mp}, \bar{c}^{(k)}_{qn}$, and $\bar{s}^{(k)}_{pq}$ in Algorithm 3 which are the modifications to avoid the zero locking. The following gives their definitions.

\[
\bar{b}^{(k)}_{mp} = \begin{cases} b^{(k)}_{mp} & \text{if } \nabla_B J(B^{(k)}, S^{(k)}, C^{(k)})_{mp} \geq 0, \\ \max(b^{(k)}_{mp}, \sigma) & \text{otherwise} \end{cases}
\]
\[
\tilde{c}^{(k)}_{qn} \equiv \begin{cases} 
    c^{(k)}_{qn} & \text{if } \nabla C J(B^{(k+1)}, S^{(k)}, C^{(k)})_{qn} \geq 0, \\
    \max(c^{(k)}_{qn}, \sigma) & \text{otherwise}
\end{cases}
\]

\[
\tilde{s}^{(k)}_{pq} \equiv \begin{cases} 
    s^{(k)}_{pq} & \text{if } \nabla S J(B^{(k+1)}, S^{(k)}, C^{(k+1)})_{pq} \geq 0, \\
    \max(s^{(k)}_{pq}, \sigma) & \text{otherwise}
\end{cases}
\]

with \(\sigma\) is a small positive number; and \(\vec{B}, \vec{C},\) and \(\vec{S}\) are matrices that contain \(\vec{b}_{mp}, \vec{c}_{qn},\) and \(\vec{s}_{pq}\) respectively. Moreover, there are also \(\delta_B, \delta_C,\) and \(\delta_S\) that play a crucial role in the convergence analysis (see the appendix). Note that Algorithm 3 is an intermediate algorithm for deriving the convergent algorithm and needs some modifications to be convergent, and Algorithm 4 shows the required modifications as suggested by Theorem 8, 9, and 10 in the appendix with step is a constant that determines how fast \(\delta^{(k)}_B, \delta^{(k)}_C,\) and \(\delta^{(k)}_S\) grow in order to satisfy the nonincreasing property.

### 4 Experimental results

The convergence of the proposed Algorithm 4 (AU-B) and the (lack of) convergence of the Algorithm 2 (MU-B) will be numerically shown in this section. As it is generally difficult to reach a stationary point in an acceptable computational time due to the slow convergence and numerical precision issues, only the nonincreasing property (or lack of it) will be investigated. The other related issues like the effects of \(\alpha\) and \(\beta\) values on the computational times of MU-B and AU-B, minimization slopes, how to choose \(\alpha\) and \(\beta\) values, and computational performances comparison between the algorithms will also be discussed. Moreover, because BNMtfF was originally designed for clustering purpose, clustering capability of the algorithms will also be investigated.

As the datasets, we use Reuters-21578 data corpus. This dataset contains 21578 documents and 135 manually created topics. Each document is assigned to one or more topics based on its content. The dataset is publicly available in two formats: SGML and XML, and is divided into 22 files with each file contains 1000 documents except the last file that contains only 578 documents. We used the XML version. We used all but the 18th file because this file is invalid both in its SGML and XML version. We used only documents that belong to exclusively one class. Further, we removed the common English stop words\(^1\), stemmed the remaining words using Porter stemmer (van Rijsbergen et al. 1980), and then removed words that belong to only one documents. We also normalized the term-by document matrix \(A\) by: \(A \leftarrow AD^{-1/2}\) where \(D = \text{diag}(A^T A e)\). The test datasets were formed by combining top 2, 4, 6, 8, 10, and 12 classes from the corpus. Table 1 summarizes the statistics of the datasets, where \#doc, \#word, \%nnz, max, and min refer to the number of documents, the number of words, percentage of nonzero entries, the number of documents in the largest cluster, and the number of documents in the smallest cluster respectively. And as the corpus is bipartite, document and word clusterings will be evaluated in a similar fashion as in the original BNMtfF paper (Ding et al. 2006).

---

\(^1\) [http://snowball.tartarus.org/algorithms/english/stop.txt](http://snowball.tartarus.org/algorithms/english/stop.txt)
A convergent algorithm for BNMtF problem in Eq. 7 (AU-B).

Algorithm 4

Initialize $B^{(0)}$, $C^{(0)}$, and $S^{(0)}$ with nonnegative matrices, and choose a small positive number for $\delta$ and an integer number for step.

for $k = 0, \ldots , K$ do

$\delta_B^{(k)} \leftarrow \delta$

repeat

$b_{mp}^{(k+1)} \leftarrow b_{mp}^{(k)} - \frac{\delta_B^{(k)} \nabla_B J(B^{(k)}, S^{(k)}, C^{(k)})_{mp}}{(B^{(k)}S^{(k)}C^{(k)})^T S^{(k)}T + \beta B^{(k)}B^{(k)}T B^{(k)} + \delta_B^{(k)}} \forall m, p$

$\delta_B^{(k)} \leftarrow \delta_B^{(k)} \times \text{step}$

until $J(B^{(k+1)}, S^{(k)}, C^{(k)}) \leq J(B^{(k)}, S^{(k)}, C^{(k)})$

$\delta_C^{(k)} \leftarrow \delta$

repeat

$c_{qn}^{(k+1)} \leftarrow c_{qn}^{(k)} - \frac{\delta_C^{(k)} \nabla_C J(B^{(k+1)}, S^{(k)}, C^{(k)})_{qn}}{(S^{(k)}B^{(k+1)T}B^{(k+1)}S^{(k)}C^{(k)})^T + \alpha C^{(k)}C^{(k)}T C^{(k)}C^{(k)} + \delta_C^{(k)}} \forall q, n$

$\delta_C^{(k)} \leftarrow \delta_C^{(k)} \times \text{step}$

until $J(B^{(k+1)}, S^{(k)}, C^{(k+1)}) \leq J(B^{(k+1)}, S^{(k)}, C^{(k)})$

$\delta_S^{(k)} \leftarrow \delta$

repeat

$s_{pq}^{(k+1)} \leftarrow s_{pq}^{(k)} - \frac{\delta_S^{(k)} \nabla_S J(B^{(k+1)}, S^{(k)}, C^{(k+1)})_{pq}}{(B^{(k+1)}S^{(k)}C^{(k+1)})^T \bar{S}^{(k)}C^{(k+1)}C^{(k+1)} + \delta_S^{(k)}} \forall p, q$

$\delta_S^{(k)} \leftarrow \delta_S^{(k)} \times \text{step}$

until $J(B^{(k+1)}, S^{(k+1)}, C^{(k+1)}) \leq J(B^{(k+1)}, S^{(k)}, C^{(k+1)})$

end for

Table 1: Statistics of the test datasets

| Dataset  | #doc | #word | %nnz | max  | min  |
|----------|------|-------|------|------|------|
| Reuters2 | 6090 | 8547  | 0.363| 3874 | 2216 |
| Reuters4 | 6797 | 9900  | 0.353| 3874 | 333  |
| Reuters6 | 7354 | 10319 | 0.347| 3874 | 269  |
| Reuters8 | 7644 | 10596 | 0.340| 3874 | 144  |
| Reuters10| 7887 | 10930 | 0.336| 3874 | 114  |
| Reuters12| 8052 | 11172 | 0.333| 3874 | 75   |
To compare performances of MU-B and AU-B with other NMF algorithms, the following algorithms are used:

- standard NMF algorithm (Lee and Seung 2000) → **LS**,
- original UNMF algorithm (Ding et al. 2006) → **D-U**,
- original BNMtF algorithm (Ding et al. 2006) → **D-B**,
- MUR based UNMF algorithm (algorithm 3 in Mirzal 2014a) → **MU-U**, and
- AUR based convergent UNMF algorithm (algorithm 4 in (Mirzal 2014a)) → **AU-U**.

All codes were written in Octave and run under Linux platform using a notebook with 1.86 GHz Intel processor and 2 GB RAM. Note that all algorithms have the same computational complexity, i.e., $K \times M \times P \times N$ where $K$ is the number of outer iterations, $M \times P$ is the size of matrix $B$ and $P \times N$ is the size of matrix $C$. However,
for AU-U and AU-B, there are additional computational complexities due to the inner iterations where for AU-U is $P \times N$ and for AU-B is $M \times P + P \times N + P \times P$ ($P$ is assumed to be equal to $Q$) given that the number of inner iterations $repeat - until$ is not too large.

### 4.1 The nonincreasing property

Figures 2, 3, 4 and 5 show graphically the nonincreasing property (or lack of it) of MU-B and AU-B. Because there are two adjustable parameters, $\alpha$ and $\beta$, we fix one parameter while studying the other. Figures 2 and 3 show the results for fixed $\beta = 1$, and Figs. 4 and 5 for fixed $\alpha = 1$. As shown, while MU-B failed to show the nonincreasing property for large $\alpha$ and $\beta$ values, AU-B preserved this property. Note that we set $\delta = \sigma = 10^{-8}$, and step = 10 in all experiments.
4.2 The effects of large regularization parameters

As large regularization parameters often make NMF algorithms unstable, and AU algorithms (AU-U and AU-B) have inner repeat-until loops that enforce the nonincreasing property, AU algorithms can be slower compared to the corresponding MU algorithms (MU-U and MU-B) when \( \alpha \) and/or \( \beta \) are large. Table 2 shows time comparison between these algorithms for Reuters4 dataset. Note that, \( \alpha \) or \( \beta \) letter is appended to the algorithm’s acronyms to indicate which parameter is being varied. For example AU-B(\( \alpha \)) denotes AU-B with fixed \( \beta \) and varied \( \alpha \). Note that the fixed parameter was set to one. As shown, the computational times of MU algorithms are independent of \( \alpha \) and \( \beta \) values, and AU algorithms became slower for some large \( \alpha \) or \( \beta \) because the inner loops were being executed. Also, there are some anomalies in the AU-B(\( \alpha \)) ones.
and AU-B(\(\beta\)) cases where for some \(\alpha\) or \(\beta\) values, execution times were unexpectedly much faster than it should be (highlighted by italic numbers). To investigate it, the number of outer iterations \#iter (corresponds to the for loop) and the number of inner iterations \#initer (the accumulation of the repeat-until loops) for AU algorithms are displayed in Table 3 (MU algorithms reach maximum predefined number of iteration for all cases: 20 iterations). As displayed, in the cases where AU algorithms performed worse than their MU counterparts, they executed the inner iterations. And when an AU algorithm performed better, its \#iter is less than \#iter of the corresponding MU algorithm and the inner iteration was not executed.

### 4.3 The minimization slopes

The minimization slopes of MU and AU algorithms are important to be studied as the algorithms can be slow to settle. As shown by Lin (2005), LS was very fast in mini-
Table 2  Time comparison (in seconds) for Reuters4 dataset

| α or β | MU-U | AU-U | MU-B(α) | AU-B(α) | MU-B(β) | AU-B(β) |
|--------|------|------|---------|---------|---------|---------|
| 0.01   | 110  | 110  | 121     | 41.1    | 122     | 27.2    |
| 0.05   | 110  | 110  | 121     | 40.9    | 121     | 40.7    |
| 0.1    | 109  | 109  | 121     | 40.8    | 121     | 41.2    |
| 0.3    | 110  | 109  | 121     | 40.4    | 121     | 41.1    |
| 0.7    | 110  | 110  | 121     | 272     | 121     | 41.2    |
| 1      | 110  | 110  | 121     | 40.8    | 121     | 273     |
| 3      | 110  | 110  | 121     | 40.4    | 121     | 41.1    |
| 7      | 110  | 110  | 121     | 40.4    | 121     | 273     |
| 10     | 110  | 110  | 121     | 40.8    | 121     | 442     |
| 30     | 109  | 110  | 121     | 272     | 121     | 525     |
| 70     | 109  | 137  | 121     | 332     | 121     | 605     |
| 100    | 110  | 232  | 121     | 382     | 121     | 579     |
| 300    | 110  | 232  | 121     | 514     | 121     | 606     |
| 700    | 110  | 461  | 121     | 607     | 121     | 606     |
| 1000   | 110  | 411  | 121     | 606     | 121     | 365     |

Table 3  #iter and #initer of AU algorithms (Reuters4)

| α or β | AU-U #iter / #initer | AU-B(α) #iter / #initer | AU-B(β) #iter / #initer |
|--------|----------------------|-------------------------|-------------------------|
| 0.01   | 20/0                 | 3/0                     | 2/0                     |
| 0.05   | 20/0                 | 3/0                     | 3/0                     |
| 0.1    | 20/0                 | 3/0                     | 3/0                     |
| 0.3    | 20/0                 | 3/0                     | 3/0                     |
| 0.7    | 20/0                 | 20/0                    | 3/0                     |
| 1      | 20/0                 | 3/0                     | 20/0                    |
| 3      | 20/0                 | 3/0                     | 3/0                     |
| 7      | 20/0                 | 3/0                     | 20/0                    |
| 10     | 20/0                 | 3/0                     | 3/0                     |
| 30     | 20/0                 | 20/0                    | 20/44                   |
| 70     | 20/7                 | 20/23                   | 20/66                   |
| 100    | 20/32                | 20/22                   | 20/88                   |
| 300    | 20/32                | 20/65                   | 20/81                   |
| 700    | 20/92                | 20/75                   | 20/88                   |
| 1000   | 20/79                | 20/90                   | 20/24                   |

mizing the objective values (errors) for some first iterations but then was increasingly much slower until settled (not necessarily reaching a stationary point). In Table 4, we display errors for some first iterations for LS, MU-U, AU-U, MU-B, and AU-B (D-U and D-B are not included because they do not have the nonincreasing property). Note that error0 is the corresponding initial error before the algorithms start running,
and error\(n\) is the error at \(n\)-th iteration. As shown, all algorithms were exceptionally good at reducing errors in the first iterations. But then, the improvements are rather negligible. Accordingly, we set maximum number of iteration to 20 for the whole experiments. Note that for this case, AU-B has converged at the third iteration.

### 4.4 Determining \(\alpha\) and \(\beta\)

In the proposed algorithms there are two parameters, \(\alpha\) and \(\beta\), that have to be learned. Reuters4 was chosen for this purpose. These parameters do not exist in the original orthogonal NMFs nor in the other orthogonal NMF algorithms (Yoo and Choi 2008, 2010; Choi 2008). However, we notice that our formulations resemble sparse NMF (Kim and Park 2007, 2008a, b) and smooth NMF (Pauca et al. 2006). As shown in Kim and Park (2007, 2008a, b), sparse and smooth NMFs usually can give good results if \(\alpha\) and/or \(\beta\) are rather small positive numbers. To determine \(\alpha\) and \(\beta\), we evaluate clustering qualities produced by our algorithms as \(\alpha\) or \(\beta\) values grow measured by the standard clustering metrics: mutual information (MI), entropy (E), purity (P), and \(F\) measure (\(F\)) where larger MI, \(F\), and P; but smaller E indicate better results. The detailed discussions on these metrics can be found in Mirzal (2014a). As shown in Fig. 6, for UNMF algorithms (MU-U and AU-U) \(\alpha = 0.1\) seems to be a good choice. For MU-B, \(\alpha = 0.1\) and \(\beta = 3\) are acceptable settings. And for AU-B, \(\alpha = 0.7\) and \(\beta = 1\) seem to be good settings. Based on this results, we decided to set \(\alpha = 0.1\) and \(\beta = 1\) for all datasets and algorithms.

### 4.5 Times, #iterations, and errors

To evaluate computational performances of the algorithms, we measure average and maximum running times, average and maximum #iterations, and average and maximum errors produced at the last iterations for 10 trials. Tables 5, 6 and 7 show the results.

As shown in the Table 5, LS generally was the fastest; however when MU-B or AU-B converged before reaching the maximum iteration (20 iterations), then these algorithms outperformed LS, except in Reuters12 where the gains from having less iterations were no longer sufficient in compensating the size of the dataset (see Table 6 for information about average and maximum #iterations). Our uni-orthogonal algorithms (MU-U and AU-U) had comparable running times with LS. MU-B was slightly slower in smaller datasets and then performed better than MU-U and AU-U in Reuters10 and Reuters12.
Fig. 6 Clustering qualities as functions of $\alpha$ or $\beta$ for Reuters4

Since AU-B usually converged before reaching the maximum iteration (see Table 6), comparison (in the worst case scenario) can be done by using the maximum running times where for Reuters4, Reuters6, Reuters10, and Reuters12 the data are available. As shown by Tables 5 and 6, AU-B was the slowest. There were also abrupt growth of the running times in Reuters10 andReuters12 for all algorithms even though as
shown in Table 1, the sizes of the datasets are only slightly larger. Figure 7 shows these abrupt changes visually.

Average and maximum errors at the corresponding last iterations are shown in Table 7. Note that the last iterations can be the maximum iteration (20) or the iterations when the algorithms have converged. Results for D-U and D-B were unsurprisingly really high as these algorithms did not minimize the corresponding objectives. Because only MU-U and AU-U and MU-B and AU-B pairs have the same objective each, we compare average errors for these pairs in Fig. 8 for each dataset. As shown, there was no significant difference between MU-U and AU-U in the average errors, and their running times were also similar, except in the larger datasets (see Fig. 7). For MU-B and AU-B, the differences in the average errors grew slightly with the sizes of the datasets with significant differences occurred in Reuters10 and Reuters12.

4.6 Document clustering

The performances of the algorithms in document clustering measured by the four metrics are shown in Table 8. In average, MU-U performed the best as measured by all metrics especially for datasets with small #clusters. Then followed by LS, AU-U, and D-U with small margins. LS was better for datasets with large #clusters. Generally, MU-U, LS, AU-U and D-U could give consistent results for varied #clusters. However, all BNMtF algorithms did not perform well for this task.

4.7 Word clustering

In some cases, the ability of clustering methods to simultaneously group similar documents with related words (co-clustering) can become an added value. Since word clustering has no reference class (ground truth), we adopted idea from (Ding et al.
Table 6  Average and maximum #iteration (10 trials)

| Data    | #iter. | LS  | D-U | D-B | MU-U | AU-U | MU-B | AU-B |
|---------|--------|-----|-----|-----|------|------|------|------|
| Reuters2| Av.    | 20  | 20  | 20  | 20   | 20   | 16.2 | 4.9  |
|         | Max.   | 20  | 20  | 20  | 20   | 20   | 6    |      |
| Reuters4| Av.    | 20  | 20  | 20  | 20   | 20   | 20   | 7.2  |
|         | Max.   | 20  | 20  | 20  | 20   | 20   | 20   |      |
| Reuters6| Av.    | 20  | 20  | 20  | 20   | 20   | 20   | 5.5  |
|         | Max.   | 20  | 20  | 20  | 20   | 20   | 20   |      |
| Reuters8| Av.    | 20  | 20  | 20  | 20   | 20   | 20   | 4    |
|         | Max.   | 20  | 20  | 20  | 20   | 20   | 4    |      |
| Reuters10| Av.  | 20  | 20  | 20  | 20   | 20   | 20   | 5.6  |
|          | Max.  | 20  | 20  | 20  | 20   | 20   | 20   |      |
| Reuters12| Av.  | 20  | 20  | 20  | 20   | 20   | 20   | 8.8  |
|          | Max.  | 20  | 20  | 20  | 20   | 20   | 20   |      |

Table 7  Average errors (10 trials) and maximum errors at the last iteration

| Data    | #iter. | LS  | D-U | D-B | MU-U | AU-U | MU-B | AU-B |
|---------|--------|-----|-----|-----|------|------|------|------|
| Reuters2| Av.    | 1.3763 | 3435.6 | 3626.5 | 1.4106 | 1.4138 | 1.7955 | 1.8021 |
|         | Max.   | 1.3854 | 3587.2 | 3867.4 | 1.4201 | 1.4230 | 1.8022 | 1.8025 |
| Reuters4| Av.    | 1.4791 | 9152.8 | 8689.0 | 1.5299 | 1.5310 | 2.0708 | 2.0962 |
|         | Max.   | 1.4855 | 9474.9 | 9297.9 | 1.5408 | 1.5402 | 2.0880 | 2.1028 |
| Reuters6| Av.    | 1.5229 | 17135 | 15,823 | 1.5844 | 1.5878 | 2.2627 | 2.2921 |
|         | Max.   | 1.5301 | 17,971 | 16,955 | 1.5884 | 1.5952 | 2.2758 | 2.2998 |
| Reuters8| Av.    | 1.5434 | 25,913 | 22,893 | 1.6215 | 1.6171 | 2.3863 | 2.4421 |
|         | Max.   | 1.5473 | 27,462 | 25,553 | 1.6342 | 1.6262 | 2.3993 | 2.4422 |
| Reuters10| Av. | 1.5696 | 34,154 | 30,518 | 1.6533 | 1.6533 | 1.8836 | 2.5673 |
|         | Max.  | 1.5801 | 35,236 | 35,152 | 1.6622 | 1.6618 | 1.9529 | 2.5718 |
| Reuters12| Av.  | 1.5727 | 42,739 | 37,038 | 1.6620 | 1.6621 | 1.8860 | 2.6551 |
|          | Max.  | 1.5815 | 44,325 | 41,940 | 1.6705 | 1.6713 | 1.9193 | 2.6697 |

(2006) where the authors proposed to create reference classes by using word frequencies: each word is assigned to a class where it appears with the highest frequency. Determining the ground truth for word clustering is a nontrivial task due to the lack of benchmarking datasets. There are some works that proposed to utilize the concept of coherence, i.e., how much the top words in a cluster are associated to each other, e.g., Salah et al. (2018). However, this approach requires large document corpora to guarantee the coherence quality and must provide a mechanism to deal with possible biases in the datasets. Thus, the results of word clustering in this section should be considered as an effort to provide a complete report about the properties of the algorithms.
5 Conclusions

We have presented a convergent algorithm for BNMtF based on a technique presented in our previous work (Mirzal 2014a). The convergence property of the algorithm is proven theoretically and its nonincreasing property is investigated numerically. As shown in the experimental results, the algorithm preserves the nonincreasing property even when the regularization parameters are large, and thus it is a stable algorithm. We
| Metrics | Data     | LS       | D-U     | D-B     | MU-U    | AU-U    | MU-B    | AU-B    |
|---------|----------|----------|---------|---------|---------|---------|---------|---------|
| MI      | Reuters2 | 0.40392  | 0.42487 | 0.36560 | 0.47507 | 0.42150 | 0.057799| 0.00087646|
|         | Reuters4 | 0.62879  | 0.61723 | 0.48007 | 0.65080 | 0.63640 | 0.32142 | 0.072621 |
|         | Reuters6 | 0.79459  | 0.81831 | 0.52498 | 0.81811 | 0.82425 | 0.37924 | 0.078201 |
|         | Reuters8 | 0.92285  | 0.90260 | 0.54534 | 0.94165 | 0.92720 | 0.48435 | 0.013518 |
|         | Reuters10| 1.0415   | 1.0275  | 0.62125 | 1.0063  | 1.0138  | 0.50980 | 0.072014 |
|         | Reuters12| 1.1326   | 1.0865  | 0.58469 | 1.1195  | 1.0821  | 0.47697 | 0.16389 |
| Average |          | 0.82071  | 0.81283 | 0.52032 | 0.83523 | 0.81754 | 0.37160 | 0.066853 |
| Entropy | Reuters2 | 0.54193  | 0.52098 | 0.58025 | 0.47078 | 0.52435 | 0.88805 | 0.94498 |
|         | Reuters4 | 0.40202  | 0.40780 | 0.47638 | 0.39102 | 0.39822 | 0.55571 | 0.68011 |
|         | Reuters6 | 0.38391  | 0.37473 | 0.48821 | 0.37481 | 0.37243 | 0.54459 | 0.66105 |
|         | Reuters8 | 0.35568  | 0.36242 | 0.48151 | 0.34941 | 0.35423 | 0.50184 | 0.65879 |
|         | Reuters10| 0.33601  | 0.34023 | 0.46233 | 0.34661 | 0.34434 | 0.49608 | 0.62786 |
|         | Reuters12| 0.31953  | 0.33239 | 0.47236 | 0.32319 | 0.33362 | 0.50241 | 0.58974 |
| Average |          | 0.38985  | 0.389760| 0.49354 | 0.37597 | 0.38787 | 0.58145 | 0.69375 |
| Metrics | Data  | LS   | D-U   | D-B   | MU-U   | AU-U   | MU-B   | AU-B   |
|---------|-------|------|-------|-------|--------|--------|--------|--------|
| Purity  | Reuters2 | 0.82154 | 0.83599 | 0.80452 | **0.85089** | 0.82507 | 0.66102 | 0.63612 |
|         | Reuters4 | 0.79417 | 0.78023 | 0.73778 | **0.80400** | 0.79704 | 0.70119 | 0.59657 |
|         | Reuters6 | 0.74510 | **0.75158** | 0.68844 | 0.74868 | 0.75069 | 0.66433 | 0.54569 |
|         | Reuters8 | **0.74906** | 0.73982 | 0.66536 | 0.74869 | 0.73987 | 0.65033 | 0.50680 |
|         | Reuters10 | 0.73120 | **0.73762** | 0.64845 | 0.72813 | 0.73330 | 0.63194 | 0.50639 |
|         | Reuters12 | 0.73877 | 0.72719 | 0.62223 | **0.74127** | 0.72340 | 0.60118 | 0.52019 |
| Average |         | **0.76331** | 0.76207 | 0.69446 | **0.77028** | 0.76156 | 0.65166 | 0.55196 |
| Fmeasure | Reuters2 | 0.81904 | 0.83234 | 0.79163 | **0.84823** | 0.82241 | 0.58237 | 0.50399 |
|         | Reuters4 | 0.56154 | 0.53754 | 0.44352 | **0.57989** | 0.54267 | 0.36917 | 0.24585 |
|         | Reuters6 | 0.46225 | 0.47714 | 0.33910 | **0.48444** | 0.47270 | 0.26372 | 0.17171 |
|         | Reuters8 | 0.40408 | 0.40554 | 0.25052 | 0.41822 | **0.42996** | 0.23904 | 0.10869 |
|         | Reuters10 | 0.38001 | **0.38041** | 0.23309 | 0.36923 | 0.35947 | 0.19552 | 0.094912 |
|         | Reuters12 | 0.35671 | **0.35811** | 0.17387 | 0.35214 | 0.34435 | 0.16401 | 0.099949 |
| Average |         | **0.49727** | 0.49851 | 0.37196 | **0.50869** | 0.49526 | 0.30231 | 0.20418 |

Bold values indicate the best performing algorithms
| Metrics | Data       | LS     | D-U    | D-B    | MU-U   | AU-U   | MU-B   | AU-B   |
|---------|------------|--------|--------|--------|--------|--------|--------|--------|
| MI      | Reuters2   | 0.15715| 0.16609| 0.12966| 0.17351| 0.14978| 0.013995| 0.00029807|
|         | Reuters4   | 0.42558| 0.39193| 0.21495| 0.42619| 0.41663| 0.11812| 0.026943|
|         | Reuters6   | 0.54112| 0.57472| 0.26971| 0.54239| 0.54828| 0.12460| 0.035309|
|         | Reuters8   | 0.63022| 0.63368| 0.29277| 0.64699| 0.65774| 0.15692| 0.0037071|
|         | Reuters10  | 0.70386| 0.73345| 0.33046| 0.66262| 0.68367| 0.025320| 0.029618|
|         | Reuters12  | 0.80111| 0.77959| 0.28412| 0.76128| 0.73517| 0.013483| 0.073478|
| Average |            | 0.54317| 0.54658| 0.25361| 0.53549| 0.53188| 0.075407| 0.028226|
| Entropy | Reuters2   | 0.76778| 0.75884| 0.79527| 0.75142| 0.77515| 0.91094| 0.92463|
|         | Reuters4   | 0.62965| 0.64647| 0.73496| 0.62934| 0.63412| 0.78338| 0.82897|
|         | Reuters6   | 0.56184| 0.54884| 0.66683| 0.56134| 0.55906| 0.72297| 0.75751|
|         | Reuters8   | 0.52006| 0.51891| 0.63255| 0.51447| 0.51089| 0.67783| 0.72890|
|         | Reuters10  | 0.50612| 0.49721| 0.61852| 0.51853| 0.51220| 0.71038| 0.70909|
|         | Reuters12  | 0.48211| 0.48811| 0.62632| 0.49322| 0.50050| 0.70181| 0.68507|
| Average |            | 0.57792| 0.57640| 0.67908| 0.57806| 0.58199| 0.75122| 0.77236|
Table 9 continued

| Metrics   | Data   | LS   | D-U   | D-B   | MU-U   | AU-U   | MU-B   | AU-B   |
|-----------|--------|------|-------|-------|--------|--------|--------|--------|
| Purity    | Reuters2 | 0.76987 | 0.77082 | 0.75378 | **0.77730** | 0.76021 | 0.67006 | 0.65988 |
|           | Reuters4 | 0.64400 | 0.62881 | 0.60566 | **0.64676** | 0.64184 | 0.55808 | 0.53116 |
|           | Reuters6 | 0.59830 | 0.61733 | 0.59494 | 0.59763 | 0.59103 | 0.52966 | 0.49661 |
|           | Reuters8 | **0.59560** | 0.58935 | 0.54296 | 0.59179 | 0.58770 | 0.50933 | 0.46499 |
|           | Reuters10 | 0.58123 | **0.60236** | 0.51576 | 0.57045 | 0.58724 | 0.44765 | 0.45395 |
|           | Reuters12 | **0.60208** | 0.59563 | 0.49555 | 0.58628 | 0.56846 | 0.43611 | 0.44882 |
| Average   | Reuters2 | 0.63185 | **0.63405** | 0.57887 | 0.62837 | 0.62274 | 0.52515 | 0.50923 |
|           | Reuters4 | 0.46891 | 0.43469 | 0.36397 | **0.48118** | 0.46180 | 0.32520 | 0.27101 |
|           | Reuters6 | 0.37490 | 0.38365 | 0.27356 | **0.38648** | 0.38026 | 0.21620 | 0.17572 |
|           | Reuters8 | 0.32488 | 0.32674 | 0.20820 | 0.33527 | **0.34251** | 0.17127 | 0.12565 |
|           | Reuters10 | 0.29864 | **0.30768** | 0.18626 | 0.28930 | 0.28573 | 0.10700 | 0.10545 |
|           | Reuters12 | **0.29116** | 0.29072 | 0.14255 | 0.27525 | 0.27380 | 0.088517 | 0.095880 |
| Average   | Reuters2 | 0.39189 | 0.38970 | 0.29365 | **0.39407** | 0.38973 | 0.23908 | 0.21224 |

Bold values indicate the best performing algorithms
then investigate other properties of the algorithm such as the effects of using different values of regularization parameters, its minimization slopes, computational times, and error per iterations.

We also investigate the document and word clustering capabilities of the algorithm and compare it to other NMF algorithms. In document clustering it is shown that both UNMF algorithms (D-U and MU-U) have comparable performances to the standard NMF algorithm (LS), and all BNMtF algorithms (D-B, MU-B, AU-B) are underperformed compared to LS. Similar patterns are also observed in word clustering cases. This indicates that imposing orthogonality constraints on both basis and coefficient matrices is not effective in improving clustering capability of NMF. However, as reported in some works (Yoo and Choi 2008; Choi 2008; Mirzal 2014a, b, 2020) depending on the datasets and application domains, imposing orthogonality constraint on one factor can sometimes improve clustering capability of NMF. Moreover, as reported in some works that take dataset structures into account, e.g., (Salah et al. 2018; Abe and Yadohisa 2019; Yoo and Choi 2010), nonnegative matrix tri-factorization can have promising clustering performances. Thus, the underperformances of the BNMtF algorithms reported here probably stem from the fact that the objective functions do not optimally capture clustering structures in the datasets. We will address the issue about how to incorporate dataset structures in our future works.

The present work shows that the technique that was previously developed for designing a convergent algorithm for UNMF, a two-block alternating least squares problem, is extendable for developing a convergent algorithm for BNMtF, a three-block alternating least squares problem. As stated in some works, e.g., (Grippo and Scandirone 2000; Henseler 2010), convergence in two-block case does not necessarily guarantee convergence in three-block case or more. It will be interesting to investigate more about underlying theoretical foundation of the technique to understand the reasons and conditions in which it is extendable. We will also address this issue in our future works.

Acknowledgements The author would like to thank the reviewers and editors for their detailed comments and suggestions that contributed greatly to the improvement of this manuscript.

A Convergence analysis

From a result in convergence analysis of block coordinate descent method (Lin 2007, 2005; Grippo and Scandirone 2000), Algorithm 3 has a convergence guarantee if the following conditions are satisfied:

1. sequence \( J(B^{(k)}, S^{(k)}, C^{(k)}) \) has nonincreasing property,
2. any limit point of sequence \( \{B^{(k)}, S^{(k)}, C^{(k)}\} \) generated by Algorithm 3 is a stationary point, and
3. sequence \( \{B^{(k)}, S^{(k)}, C^{(k)}\} \) has at least one limit point.

Because Algorithm 3 uses the alternating strategy, sequences \( J(B^{(k)}) \), \( J(C^{(k)}) \), and \( J(S^{(k)}) \) can be analyzed separately (Lee and Seung 2000; Lin 2007). And because
update rule for $B^{(k)}$ in Eq. 9 is similar to the update rule for $C^{(k)}$ in Eq. 10, it suffices to prove nonincreasing property of one of them.

A.1 Nonincreasing property of $J(\mathbf{B}^{(k)})$

By using auxiliary function approach (Lee and Seung 2000), nonincreasing property of $J(\mathbf{B}^{(k)})$ can be proven if the following statement is true:

$$ J(\mathbf{B}^{(k+1)}) = G(\mathbf{B}^{(k+1)}, \mathbf{B}^{(k)}) \leq G(\mathbf{B}^{(k+1)}, \mathbf{B}^{(k)}) \leq G(\mathbf{B}^{(k)}, \mathbf{B}^{(k)}) = J(\mathbf{B}^{(k)}). $$

To define $G$, let us rearrange $\mathbf{B}$ into:

$$ \mathbf{b}^T \equiv \begin{bmatrix} \mathbf{b}_1^T \\ \mathbf{b}_2^T \\ \vdots \\ \mathbf{b}_M^T \end{bmatrix} \in \mathbb{R}^{MP \times M}, $$

where $\mathbf{b}_m$ is the $m$-th row of $\mathbf{B}$. And also let us define:

$$ \nabla_{\mathbf{b}^T} \mathbf{J}(\mathbf{B}^{(k)})^T = \begin{bmatrix} \nabla_{\mathbf{b}_1} \mathbf{J}(\mathbf{B}^{(k)})^T \\ \nabla_{\mathbf{b}_2} \mathbf{J}(\mathbf{B}^{(k)})^T \\ \vdots \\ \nabla_{\mathbf{b}_M} \mathbf{J}(\mathbf{B}^{(k)})^T \end{bmatrix} \in \mathbb{R}^{MP \times M}, $$

where $\nabla_{\mathbf{b}} \mathbf{J}(\mathbf{B}^{(k)})_m$ is the $m$-th row of $\nabla_{\mathbf{b}} \mathbf{J}(\mathbf{B}^{(k)})$. Then we define:

$$ \mathbf{D} \equiv \text{diag}(\mathbf{D}^1, \ldots, \mathbf{D}^M) \in \mathbb{R}^{MP \times MP}, $$

where $\mathbf{D}^m$ is a diagonal matrix with its diagonal entries defined as:

$$ d_{pp}^m = \begin{cases} \frac{\mathbf{B}_p^T \mathbf{s}^{(k)} \mathbf{c}^{(k)} \mathbf{s}^{(k)} \mathbf{c}^{(k)} \mathbf{s}^{(k)} T + \beta \mathbf{B}_p^T \mathbf{B}_p^T \mathbf{b}_p}{\mathbf{b}_m^T} + \delta_k & \text{if } p \in \mathcal{I}_m \\ * & \text{if } p \notin \mathcal{I}_m \end{cases} $$
with

\[ \mathcal{I}_m \equiv \{ p \mid p_{mp}^{(k)} > 0, \, \nabla_{B_J} (B^{(k)})_{mp} \neq 0, \text{ or} \]

\[ p_{mp}^{(k)} = 0, \, \nabla_{B_J} (B^{(k)})_{mp} < 0 \} \]

denotes the set of non-KKT indices in \( m \)-th row of \( B^{(k)} \), and the symbol \( \star \) is defined so that \( \star \equiv 0 \) and \( \star^{-1} \equiv 0 \).

Then, the auxiliary function \( \mathcal{G} \) can be defined as:

\[ \mathcal{G}(B^T, B^{(k)}T) \equiv \mathcal{J}(B^{(k)}T) + \text{tr} \left\{ (B - B^{(k)}) \nabla_{B_J} \mathcal{J}(B^{(k)}T) \right\} \]
\[ + \frac{1}{2} \text{tr} \left\{ (B - B^{(k)}) D (B - B^{(k)})^T \right\}. \]  

(12)

Note that \( \mathcal{J} \) and \( \mathcal{G} \) are equivalent to \( \mathcal{J} \) and \( \mathcal{G} \) with \( B \) is rearranged into \( B^T \), and other variables are reordered accordingly. And:

\[ \nabla_{B_J} \mathcal{G}(B^T, B^{(k)}T) = D(B - B^{(k)})^T + \nabla_{B_J} \mathcal{J}(B^{(k)}T). \]

By definition, \( D \) is positive definite for all \( B^{(k)} \) not satisfy the KKT conditions, so \( \mathcal{G}(B^T, B^{(k)}T) \) is a strict convex function, and consequently has a unique minimum.

\[ D(B - B^{(k)})^T + \nabla_{B_J} \mathcal{J}(B^{(k)}T) = 0, \]
\[ B^T = B^{(k)}T - D^{-1} \nabla_{B_J} \mathcal{J}(B^{(k)}T), \]  

(13)

which is exactly the update rule for \( B^{(k)} \).

By using the Taylor series expansion, \( \mathcal{J}(B^T) \) can also be written as:

\[ \mathcal{J}(B^T) = \mathcal{J}(B^{(k)}T) + \text{tr} \left\{ (B - B^{(k)}) \nabla_{B_J} \mathcal{J}(B^{(k)}T) \right\} \]
\[ + \frac{1}{2} \text{tr} \left\{ (B - B^{(k)}) \nabla_{B_J}^2 \mathcal{J}(B^{(k)}) (B - B^{(k)})^T \right\} + \varepsilon^{(k)}_B, \]  

(14)

where

\[ \varepsilon^{(k)}_B = \frac{1}{6} \text{tr} \left\{ (B - B^{(k)})(6B^2 B^{(k)T}) (B - B^{(k)}) (B - B^{(k)})^T \right\} \]
\[ + \frac{1}{24} \text{tr} \left\{ (B - B^{(k)})(B - B^{(k)})^T (6B^2 I) (B - B^{(k)}) (B - B^{(k)})^T \right\} \]

and

\[ \nabla_{B_J}^2 \mathcal{J}(B^{(k)}) \equiv \begin{bmatrix} \nabla_{B_J}^2 \mathcal{J}(B^{(k)}) & \vdots & \nabla_{B_J}^2 \mathcal{J}(B^{(k)}) \end{bmatrix} \in \mathbb{R}^{MP \times MP} \]
with \( \nabla^2 J(B^{(k)}) \) components are arranged along its diagonal area (there are \( M \) components).

To show the nonincreasing property of \( J(B^{(k)}) \), the following statements must be proven:

1. \( \mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) = \mathcal{J}(\mathcal{B}^T) \),
2. \( \mathcal{G}(\mathcal{B}^{(k)}T, \mathcal{B}^{(k)})T = \mathcal{J}(\mathcal{B}^{(k)}T) \),
3. \( \mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) \leq \mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) \), and
4. \( \mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) \leq \mathcal{G}(\mathcal{B}^{(k)}T, \mathcal{B}^{(k)}T) \).

The first and second statements are obvious from the definition of \( \mathcal{G} \) in Eq. 12, the third and the fourth statements will be proven in Theorem 1 and 2, and the boundedness of \( \mathcal{B}^{(k)}, \mathcal{C}^{(k)}, \) and \( \mathcal{S}^{(k)} \) will be proven in Theorem 10.

**Theorem 1** Given sufficiently large \( \delta^{(k)}_B \) and the boundedness of \( \mathcal{B}^{(k)}, \mathcal{C}^{(k)}, \) and \( \mathcal{S}^{(k)} \), then it can be shown that \( \mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) \leq \mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) \). Moreover, if and only if \( B^{(k)} \) satisfies the KKT conditions, then the equality holds.

**Proof** By subtracting Eq. 12 from Eq. 14, we get:

\[
\mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) - \mathcal{G}(\mathcal{B}^T, \mathcal{B}^{(k)}T) = \frac{1}{2} \text{tr} \left\{ (\mathcal{B} - \mathcal{B}^{(k)})(D - \nabla^2 J(B^{(k)}))(\mathcal{B} - \mathcal{B}^{(k)})^T \right\} - \varepsilon^{(k)}_B
\]

\[
= \frac{1}{2} \sum_{m=1}^{M} \left[ (b_m - b_m^{(k)})(D^m - \nabla^2 J(B^{(k)}))(b_m - b_m^{(k)})^T \right] - \varepsilon^{(k)}_B. \tag{15}
\]

Let \( v_m^T = b_m - v_m^{(k)} \), then:

\[
v_m^T(D^m - \nabla^2 J(B^{(k)}))v_m = v_m^T(D^m + \beta I - (S^{(k)}C^{(k)}C^{(k)}T + 3\beta B^{(k)}T B^{(k)}))v_m = v_m^T(\hat{D}^m + \delta^{(k)}_B \hat{D}^m + \beta I - (S^{(k)}C^{(k)}C^{(k)}T S^{(k)}T + 3\beta B^{(k)}T B^{(k)}))v_m,
\]

where \( \hat{D}^m \) and \( \delta^{(k)}_B \hat{D}^m \) are diagonal matrices that summed up to \( D^m \), with

\[
d_{pp}^m = \begin{cases} \left( \hat{B}^{(k)}S^{(k)}C^{(k)}C^{(k)}T + \beta \hat{B}^{(k)}T \hat{B}^{(k)} \right)_{pp}^{(k)} & \text{if } p \in \mathcal{I}_m \\
* & \text{if } p \notin \mathcal{I}_m \end{cases}
\]

\[
\hat{d}_{pp}^m = \begin{cases} \frac{\hat{b}_{mp}^{(k)}}{b_{mp}} & \text{if } p \in \mathcal{I}_m \\
* & \text{if } p \notin \mathcal{I}_m \end{cases}
\]
Accordingly,
\[
\mathcal{G}(B^T, B^{(k)}T) - \mathcal{G}(B^T, B^T) = \frac{1}{2} \sum_{m=1}^{M} \left\{ \sum_{p=1}^{P} v_{mp}^2 \delta_{pp}^m + \delta_B^{(k)} \sum_{p=1}^{P} v_{mp}^2 \delta_{pp}^m + \beta \sum_{p=1}^{P} v_{mp}^2 \right\} \\
- \frac{1}{2} \sum_{m=1}^{M} \left( S^{(k)} C^{(k)} C^{(k)T} S^{(k)T} + 3 \beta B^{(k)} T B^{(k)} \right) v_m - \tilde{\varepsilon}_B^{(k)}. \tag{16}
\]

As shown, with the boundedness of \(B^{(k)}, C^{(k)},\) and \(S^{(k)}\) and by sufficiently large \(\delta_B^{(k)}\), \(\mathcal{G}(B^T, B^{(k)}T) \leq \mathcal{G}(B^T, B^{(k)}T)\) can be guaranteed.

Next we prove the second statement of the theorem. By Eq. 15, if \(B^{(k)}\) satisfies the KKT conditions, then the equality will hold. And by Eq. 16, since \(\delta_B^{(k)}\) is a variable, the equality will hold if and only if \(B = B^{(k)}\), which by the update rule in Eq. 9 will happen if and only if \(B^{(k)}\) satisfies the KKT conditions. This completes the proof. □

**Theorem 2** \(\mathcal{G}(B^T, B^{(k)}T) \leq \mathcal{G}(B^{(k)}T, B^{(k)}T)\). Moreover if and only if \(B^{(k)}\) satisfies the KKT conditions in Eqs. 8, then the equality holds.

**Proof**
\[
\mathcal{G}(B^{(k)}T, B^{(k)}T) - \mathcal{G}(B^T, B^{(k)}T) = - \text{tr} \left\{ (B - B^{(k)}) \nabla_B \mathcal{J}(B^{(k)}T) \right\} \\
- \frac{1}{2} \text{tr} \left\{ (B - B^{(k)}) D (B - B^{(k)}T) \right\}.
\]

Substituting Eq. 13 into the above equation, we get:
\[
\mathcal{G}(B^{(k)}T, B^{(k)}T) - \mathcal{G}(B^T, B^{(k)}T) = \frac{1}{2} \text{tr} \left\{ (B - B^{(k)}) D (B - B^{(k)}T) \right\}.
\]

By the fact that \(D\) is positive definite for all \(B \neq B^{(k)}\) and positive semi-definite if and only if \(B = B^{(k)}\), it is proven that \(\mathcal{G}(B^T, B^{(k)}T) \leq \mathcal{G}(B^{(k)}T, B^{(k)}T)\) with the equality happens if and only if \(B^{(k)}\) satisfies the KKT conditions.

The following theorem summarizes the above results.

**Theorem 3** Given sufficiently large \(\delta_B^{(k)}\) and the boundedness of \(B^{(k)}, C^{(k)},\) and \(S^{(k)}\), \(J(B^{(k+1)}) \leq J(B^{(k)}) \ \forall k \geq 0\) under update rule Eq. 9 with the equality holds if and only if \(B^{(k)}\) satisfies the KKT conditions in Eq. 8.

A.2 Nonincreasing property of \(J(C^{(k)})\)

**Theorem 4** Given sufficiently large \(\delta_C^{(k)}\) and the boundedness of \(B^{(k)}, C^{(k)},\) and \(S^{(k)}\), \(J(C^{(k+1)}) \leq J(C^{(k)}) \ \forall k \geq 0\) under update rule Eq. 10 with the equality holds if and only if \(C^{(k)}\) satisfies the KKT conditions in Eq. 8.
Proof This theorem can be proven similarly as in $J(B^{(k)})$ case. □

A.3 Nonincreasing property of $J(S^{(k)})$

Next we prove the nonincreasing property of $J(S^{(k)})$, i.e., $J(S^{(k+1)}) \leq J(S^{(k)}) \forall k \geq 0$.

By using the auxiliary function approach, the nonincreasing property of $J(S^{(k)})$ can be proven by showing that:

$$J(S^{(k+1)}) = G(S^{(k+1)}, S^{(k+1)}) \leq G(S^{(k+1)}, S^{(k)}) \leq G(S^{(k)}, S^{(k)}) = J(S^{(k)}).$$

To define $G$, $S$ is rearranged into:

$$\mathcal{G} \equiv \begin{bmatrix} s_1 \\
  s_2 \\
  \vdots \\
  s_Q \end{bmatrix} \in \mathbb{R}^{PQ \times Q},$$

where $s_q$ is the $q$-th column of $S$. And also let us define:

$$\nabla \mathcal{G} \mathcal{J}(S^{(k)}) \equiv \begin{bmatrix} \nabla s_1 \mathcal{J}(S^{(k)}) \\
  \nabla s_2 \mathcal{J}(S^{(k)}) \\
  \vdots \\
  \nabla s_Q \mathcal{J}(S^{(k)}) \end{bmatrix} \in \mathbb{R}^{PQ \times Q},$$

where $\nabla s_q \mathcal{J}(S^{(k)})$ is the $q$-th column of $\nabla s_j(S^{(k)})$. And:

$$D \equiv \text{diag}(D^1, \ldots, D^Q) \in \mathbb{R}^{PQ \times PQ},$$

where $D^q$ is a diagonal matrix with its diagonal entries defined as:

$$d_{pp}^q \equiv \begin{cases} 
  \frac{(B^{(k+1)^T} B^{(k+1)} S^{(k)} C^{(k+1)} C^{(k+1)^T})_{pq} + \delta_{pq}^{(k)}}{\delta_{pq}^{(k)}} & \text{if } p \in \mathcal{I}_q \\
  \star & \text{if } p \notin \mathcal{I}_q 
\end{cases}$$

with

$$\mathcal{I}_q \equiv \{ p | s_{pq}^{(k)} > 0, \nabla_S J(S^{(k)})_{pq} \neq 0, \text{ or } s_{pq}^{(k)} = 0, \nabla_S J(S^{(k)})_{pq} < 0 \}$$

is the set of non-KKT indices in $q$-th column of $S^{(k)}$, and $\star$ is defined as before.
Then, the auxiliary function $\mathcal{G}$ can be written as:

$$
\mathcal{G}(\mathcal{S}, \mathcal{S}(k)) \equiv \mathcal{J}(\mathcal{S}(k)) + \text{tr} \left\{ (\mathcal{S} - \mathcal{S}(k))^T \nabla_{\mathcal{S}} \mathcal{J}(\mathcal{S}(k)) \right\} + \frac{1}{2} \text{tr} \left\{ (\mathcal{S} - \mathcal{S}(k))^T \mathcal{D}(\mathcal{S} - \mathcal{S}(k)) \right\}.
$$

(17)

Also:

$$
\nabla_{\mathcal{S}} \mathcal{G}(\mathcal{S}, \mathcal{S}(k)) = \mathcal{D}(\mathcal{S} - \mathcal{S}(k)) + \nabla_{\mathcal{S}} \mathcal{J}(\mathcal{S}(k)).
$$

Since $\mathcal{G}(\mathcal{S}, \mathcal{S}(k))$ is a strict convex function, it has a unique minimum.

$$
\mathcal{D}(\mathcal{S} - \mathcal{S}(k)) + \nabla_{\mathcal{S}} \mathcal{J}(\mathcal{S}(k)) = 0,
$$

$$
\mathcal{S} = \mathcal{S}(k) - \mathcal{D}^{-1} \nabla_{\mathcal{S}} \mathcal{J}(\mathcal{S}(k)),
$$

(18)

which is exactly the update rule for $\mathcal{S}$ in Eq. 11.

By using the Taylor series, an alternative formulation for $\mathcal{J}(\mathcal{S})$ can be written as:

$$
\mathcal{J}(\mathcal{S}) = \mathcal{J}(\mathcal{S}(k)) + \text{tr} \left\{ (\mathcal{S} - \mathcal{S}(k))^T \nabla_{\mathcal{S}} \mathcal{J}(\mathcal{S}(k)) \right\} + \frac{1}{2} \text{tr} \left\{ (\mathcal{S} - \mathcal{S}(k))^T \nabla_{\mathcal{S}}^2 \mathcal{J}(\mathcal{S}(k)) (\mathcal{S} - \mathcal{S}(k)) \right\}
$$

(19)

where

$$
\nabla_{\mathcal{S}}^2 \mathcal{J}(\mathcal{S}(k)) \equiv \begin{bmatrix}
\nabla_{\mathcal{S}}^2 \mathcal{J}(\mathcal{S}(k)) \\
\vdots \\
\nabla_{\mathcal{S}}^2 \mathcal{J}(\mathcal{S}(k))
\end{bmatrix} \in \mathbb{R}^{PQ \times PQ}
$$

with $\nabla_{\mathcal{S}}^2 \mathcal{J}(\mathcal{S}(k))$ components are arranged along its diagonal area (there are $Q$ components).

For $\mathcal{G}$ to be the auxiliary function, we must prove:

1. $\mathcal{G}(\mathcal{S}, \mathcal{S}) = \mathcal{J}(\mathcal{S})$,
2. $\mathcal{G}(\mathcal{S}(k), \mathcal{S}(k)) = \mathcal{J}(\mathcal{S}(k))$,
3. $\mathcal{G}(\mathcal{S}, \mathcal{S}) \leq \mathcal{G}(\mathcal{S}, \mathcal{S}(k))$, and
4. $\mathcal{G}(\mathcal{S}, \mathcal{S}(k)) \leq \mathcal{G}(\mathcal{S}(k), \mathcal{S}(k))$.

The first and second are clear from the definition of $\mathcal{G}$ in Eq. 17, the third and the fourth statements are proven below.

**Theorem 5** Given sufficiently large $\delta_{\mathcal{S}}(k)$ and the boundedness of $\mathcal{B}(k)$, $\mathcal{C}(k)$, and $\mathcal{S}(k)$, then it can be shown that $\mathcal{G}(\mathcal{S}, \mathcal{S}) \leq \mathcal{G}(\mathcal{S}, \mathcal{S}(k))$. Moreover, if and only if $\mathcal{S}(k)$ satisfies the KKT conditions, then the equality holds.
Theorem 6

Let $v_q = s_q - s_q^{(k)}$, then:

$$
v_q^T (D^q - \nabla_S J(S^{(k)})) v_q = v_q^T (D^q - (B^{(k+1)} B^{(k+1)T} C^{(k+1)} C^{(k+1)T})) v_q
$$

$$= v_q^T (\tilde{D}^q + \delta_S^{(k)} \delta^q - (B^{(k+1)} B^{(k+1)} C^{(k+1)} C^{(k+1)T})) v_q,$$

where $\tilde{D}^q$ and $\delta_S^{(k)} \delta^q$ are diagonal matrices that summed up to $D^q$, with

$$\tilde{d}_{pp}^q \equiv \frac{(B^{(k+1)} B^{(k+1)} \tilde{S}^{(k)} C^{(k+1)} C^{(k+1)T})_{pq}}{s_{pq}^{(k)}} \text{ if } p \in I_q \text{ and } d_{pp}^q \equiv \frac{1}{s_{pq}^{(k)}} \text{ if } p \in I_q,$$

$$\text{ and } d_{pp}^q \equiv \ast \text{ if } p \notin I_q.$$

Accordingly,

$$\mathcal{G}(S, S^{(k)}) - M(S) = \frac{1}{2} \sum_{q=1}^Q \left[ \sum_{p=1}^P v_{pq}^2 d_{pp}^q + \delta_S^{(k)} \sum_{p=1}^P v_{pq}^2 \delta_{pp}^q \right]
$$

$$- \frac{1}{2} \sum_{q=1}^Q v_q^T (B^{(k+1)} B^{(k+1)} C^{(k+1)} C^{(k+1)T}) v_q.$$

As shown, with the boundedness of $B^{(k)}$, $C^{(k)}$, and $S^{(k)}$, and by sufficiently large $\delta_S^{(k)}$, $\mathcal{G}(S, S) \leq \mathcal{G}(S^{(k)}, S^{(k)})$ can be guaranteed.

Next we prove the second statement of the theorem. By Eq. 20 if $S^{(k)}$ satisfies the KKT conditions, then the equality will hold. And by Eq. 21, since $\delta_S^{(k)}$ is a variable, the equality will hold if and only if $S = S^{(k)}$ which by the update rule in Eq. 11 will happen if and only if $S^{(k)}$ satisfies the KKT conditions. This completes the proof. □

Theorem 6

$\mathcal{G}(S, S^{(k)}) \leq \mathcal{G}(S^{(k)}, S^{(k)})$. Moreover if and only if $S^{(k)}$ satisfies the KKT conditions, then $\mathcal{G}(S, S^{(k)}) = \mathcal{G}(S^{(k)}, S^{(k)}).

Proof

Substituting Eq. 18 into the above equation, we get:

$$\mathcal{G}(S^{(k)}, S^{(k)}) - \mathcal{G}(S, S^{(k)}) = \frac{1}{2} \text{tr} \left\{ (S - S^{(k)})^T D (S - S^{(k)}) \right\} \geq 0.$$
By the fact that $D$ is positive definite for all $G \neq G^{(k)}$ and positive semi-definite if and only if $G = G^{(k)}$, it is proven that $G (G, G^{(k)}) \leq G (G^{(k)}, G^{(k)})$ with the equality holds if and only if $S^{(k)}$ satisfies the KKT conditions.

The following theorem summarizes the above results.

**Theorem 7** Given sufficiently large $\delta_S^{(k)}$ and the boundedness of $B^{(k)}$, $C^{(k)}$, and $S^{(k)}$, $J(S^{k+1}) \leq J(S^{(k)}) \forall k \geq 0$ under update rule Eq. 11 with the equality holds if and only if $S^{(k)}$ satisfies the KKT conditions in Eq. 8.

**A.4 The nonincreasing property of sequence $J(B^{(k)}, S^{(k)}, C^{(k)})$**

As stated in the beginning of the appendix, this property is the first point needs to be proven in order to show the convergence of Algorithm 3.

**Theorem 8** Given sufficiently large $\delta_B^{(k)}$, $\delta_C^{(k)}$, and $\delta_S^{(k)}$, and the boundedness of $B^{(k)}$, $C^{(k)}$, and $S^{(k)}$, $J(B^{(k+1)}, S^{(k+1)}, C^{(k+1)}) \leq J(B^{(k+1)}, S^{(k)}, C^{(k+1)}) \leq J(B^{(k+1)}, S^{(k)}, C^{(k)}) \leq J(B^{(k)}, S^{(k)}, C^{(k)})$ for $\forall k \geq 0$ under update rules in Algorithm 3 with the equalities happen if and only if $(B^{(k)}, S^{(k)}, C^{(k)})$ satisfies the KKT optimality conditions.

**Proof** This statement can be proven by combining the results in theorems 3, 4, and 7.

**A.5 Limit points of sequence $\{B^{(k)}, S^{(k)}, C^{(k)}\}$**

**Theorem 9** Given sufficiently large $\delta_B^{(k)}$, $\delta_C^{(k)}$, and $\delta_S^{(k)}$, and the boundedness of $B^{(k)}$, $C^{(k)}$, and $S^{(k)}$, it can be shown that any limit point of sequence $\{B^{(k)}, S^{(k)}, C^{(k)}\}$ generated by Algorithm 3 is a stationary point.

**Proof** By Theorem 8, Algorithm 3 produces strictly decreasing sequence $J(B^{(k)}, S^{(k)}, C^{(k)})$ until reaching a point that satisfies the KKT conditions. Because $J(B^{(k)}, S^{(k)}, C^{(k)}) \geq 0$, this sequence is bounded and thus converges. And by the update rules in Algorithm 3, after a point satisfies the KKT conditions, the algorithm will stop updating $(B^{(k)}, S^{(k)}, C^{(k)})$, i.e., $B^{(k+1)} = B^{(k)}$, $C^{(k+1)} = C^{(k)}$, and $S^{(k+1)} = S^{(k)} \forall k \geq \star$ where $\star$ denotes the iteration number where the KKT conditions have been satisfied. This completes the proof.

**Theorem 10** The sequence $\{B^{(k)}, S^{(k)}, C^{(k)}\}$ has at least one limit point.

**Proof** Due to the result in Theorem 9, it suffices to prove that sequence $\{B^{(k)}, S^{(k)}, C^{(k)}\}$ is in a closed and bounded set. By the objective in Eq. 7 it is clear that $\{B^{(k)}, S^{(k)}, C^{(k)}\}$ has a nonnegative lower bound. Thus, only upper-boundedness of this sequence needs to be proven. If there exists $l$ such that $\lim p_{mp}^{(l)} \rightarrow \infty$ or $\lim c_{qn}^{(l)} \rightarrow \infty$, then $\lim J \rightarrow \infty > J(B^{(0)}, S^{(0)}, C^{(0)})$ which violates Theorem 8. And if $\{S^{(k)}\}$ is not

\[ \text{ Springer} \]
upper-bounded, then there exists \( l \) such that \( \lim_{s_{pq} \to \infty} s_{pq}^{(l)} < s_{pq}^{(l+1)} \). Due to Theorem 8, \( J(B^{(k)}, S^{(k)}, C^{(k)}) \) is upper-bounded, then this means that either \( b_{mp}^{(l)} \) for \( \forall m \) or \( c_{qn}^{(l)} \) for \( \forall n \) must be equal to zero. If \( b_{mp}^{(l)} = 0 \) for \( \forall m \), then \( \nabla_S J_{pq} = 0 \) for \( \forall q \), so that \( s_{pq}^{(l+1)} = s_{pq}^{(l)} \). And if \( c_{qn}^{(l)} = 0 \) for \( \forall n \), then \( \nabla_S J_{pq} = 0 \) for \( \forall p \), so that \( s_{pq}^{(l+1)} = s_{pq}^{(l)} \). Both cases contradict the condition for unboundedness of \( \{S^{(l)}\} \). Thus, \( \{S^{(l)}\} \) is also upper-bounded. □

References

Abe H, Yadohisa H (2019) Orthogonal nonnegative matrix tri-factorization based on tweedie distributions. Adv Data Anal Classif 13(4):825–853

Berry M, Browne M, Langville A, Pauca V, Plemmons R (2006) Algorithms and applications for approximate nonnegative matrix factorization. In: Computational statistics and data analysis, pp 155–173

Choi S (2008) Algorithms for orthogonal nonnegative matrix factorization. In: Proceedings of IEEE international joint conference on neural networks, pp 1828–1832

Ding C, Li T, Peng W, Park H (2006) Orthogonal nonnegative matrix tri-factorizations for clustering. In: Proceedings of 12th ACM SIGKDD international conference on knowledge discovery and data mining, pp 126–135

Grippo L, Sciandrone M (2000) On the convergence of the block nonlinear gauss-seidel method under convex constraints. Oper Res Lett 26(3):127–136

Henseler J (2010) On the convergence of the partial least squares path modeling algorithm. Comput Stat 25(1):107–120

Kim D, Sra S, Dhillon I (2007) Fast newton-type methods for the least squares nonnegative matrix approximation problem. In: Proceedings of SIAM conference in data mining, pp 343–354

Kim D, Sra S, Dhillon I (2008) Fast projection-based methods for the least squares nonnegative matrix approximation problem. Stat Anal Data Min 1:38–51

Kim H, Park H (2007) Sparse non-negative matrix factorizations via alternating non-negativity constrained least squares for microarray data analysis. Bioinformatics 23(12):1495–1502

Kim H, Park H (2008) Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method. SIAM J Matrix Anal Appl 30(2):713–730

Kim J, Park H (2008) Sparse nonnegative matrix factorization for clustering. Technical report. CSE Technical Reports, Georgia Institute of Technology

Kim J, Park H (2008) Toward faster nonnegative matrix factorization: a new algorithm and comparisons. In: Proceedings of the 2008 eighth IEEE international conference on data mining, pp 353–362

Lee D, Seung H (1999) Learning the parts of objects by non-negative matrix factorization. Nature 401:788–791

Lee D, Seung H (2000) Algorithms for non-negative matrix factorization. In: Proceedings of advances in neural information processing systems, pp 556–562

Lin C (2005) Projected gradient methods for non-negative matrix factorization. Technical Report ISSTECH-95-013, Department of CS, National Taiwan University

Lin C (2007) On the convergence of multiplicative update algorithms for nonnegative matrix factorization. IEEE Trans Neural Netw 18(6):1589–1596

Mirzal A (2014a) A convergent algorithm for orthogonal nonnegative matrix factorization. J Comput Appl Math 260:149–166

Mirzal A (2014b) Nonparametric orthogonal nmf and its application in cancer clustering. In: Proceedings of the first international conference on advanced data and information engineering (DaEng-2013). Springer, pp 177–184

Mirzal A (2014c) Nonparametric tikhonov regularized nmf and its application in cancer clustering. IEEE/ACM Trans Comput Biol Bioinform 11(6):1208–1217

Mirzal A (2020) Statistical analysis of clustering performances of nmf, spectral clustering, and k-means. In: 2020 International conference on computer and information sciences (to appear)

Pauca V, Piper J, Plemmons R (2006) Nonnegative matrix factorization for spectral data analysis. Linear Algib Appl 416(1):29–47
van Rijsbergen C, Robertson S, Porter M (1980) New models in probabilistic information retrieval. Technical Report, British Library Research and Development
Salah A, Ailem M, Nadif M (2018) Word co-occurrence regularized non-negative matrix tri-factorization for text data co-clustering. In: Thirty-second AAAI conference on artificial intelligence
Yoo J, Choi S (2008) Orthogonal nonnegative matrix factorization: multiplicative updates on stiefel manifolds. In: Proceedings of the 9th international conference on intelligent data engineering and automated learning. Springer-Verlag, pp 140–147
Yoo J, Choi S (2010) Orthogonal nonnegative matrix tri-factorization for co-clustering: multiplicative updates on stiefel manifolds. Inf Process Manage 46(5):559–570
Zhao R, Tan VY (2017) A unified convergence analysis of the multiplicative update algorithm for regularized nonnegative matrix factorization. IEEE Trans Sig Process 66(1):129–138

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.