Mach Learn (2016) 103:285–306
DOI 10.1007/s10994-016-5553-0

A column-wise update algorithm for nonnegative matrix factorization in Bregman divergence with an orthogonal constraint

Keigo Kimura[1] · **Mineichi Kudo**[1] · **Yuzuru Tanaka**[1]

Received: 23 March 2015 / Accepted: 23 February 2016 / Published online: 11 March 2016

Abstract Recently orthogonal nonnegative matrix factorization (ONMF), imposing an orthogonal constraint into NMF, has been attracting a great deal of attention. ONMF is more appropriate than standard NMF for a clustering task because the constrained matrix can be considered as an indicator matrix. Several iterative ONMF algorithms have been proposed, but they suffer from slow convergence because of their matrix-wise updating. In this paper, therefore, a column-wise update algorithm is proposed for speeding up ONMF. To make the idea possible, we transform the matrix-based orthogonal constraint into a set of column-wise orthogonal constraints. The algorithm is stated first with the Frobenius norm and then with Bregman divergence, both for measuring the degree of approximation. Experiments on one artificial and six real-life datasets showed that the proposed algorithms converge faster than the other conventional ONMF algorithms, more than four times in the best cases, due to their smaller numbers of iterations.

Keywords Orthogonal nonnegative matrix factorization · Orthogonal Factorization · Bregman Divergence · Column-wise Update

Editors: Hang Li, Dinh Phung, Tru Cao, Tu-Bao Ho, and Zhi-Hua Zhou.

✉ Keigo Kimura
kkimura@main.ist.hokudai.ac.jp

Mineichi Kudo
mine@main.ist.hokudai.ac.jp

Yuzuru Tanaka
tanaka@meme.hokudai.ac.jp

[1] Graduate School of Information Science and Technology, Hokkaido University, Sapporo 060-0814, Japan

1 Introduction

Orthogonal nonnegative matrix factorization (ONMF), first proposed by Ding et al. (2006), factorizes a nonnegative matrix into two nonnegative matrices under a one-sided orthogonal constraint imposed on the first *factor* matrix. That is, ONMF is a minimization problem:

$$\begin{aligned} &\min_{\mathbf{F},\mathbf{G}} \|\mathbf{X} - \mathbf{F}\mathbf{G}^T\|_F^2, \\ &\text{subject to } \mathbf{F} \geq 0,\ \mathbf{G} \geq 0,\ \mathbf{F}^T\mathbf{F} = \mathbf{I}, \end{aligned} \tag{1}$$

where $\mathbf{X} \in \mathbb{R}^{M\times N}$, $\mathbf{F} \in \mathbb{R}^{M\times J}$, $\mathbf{G} \in \mathbb{R}^{N\times J}$ ($J \ll N, M$) and $\mathbf{I}$ is the identity matrix. In addition, T denotes the transpose and $\|\cdot\|_F^2$ denotes the squared Frobenius norm (the sum of squared elements). In this formulation, $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ is imposed as a condition, but the strict application of both nonnegativity and orthogonality is too strong. In fact, it yields a subset of orthonormal vectors in the standard basis. Therefore, in a practical sense, the optimization problem is stated as

$$\min_{\mathbf{F},\mathbf{G}} \|\mathbf{X} - \mathbf{F}\mathbf{G}^T\|_F^2 + \lambda\|\mathbf{F}^T\mathbf{F} - \mathbf{I}\| \tag{2}$$

with a positive coefficient λ. This corresponds to a Lagrangian formulation, as will be shown in the following section.

To the best of the authors' knowledge, conventional algorithms for solving ONMF problems are all based on matrix-wise alternating block coordinate descent. However, it is known that matrix-wise update algorithms require a relatively large number of iterations to converge. This is because those algorithms do not solve each conditional matrix-wise problem optimally (Cichocki and Anh-Huy 2009; Kim and Park 2011). In NMF without the orthogonal constraint, some state-of-the-art algorithms update $\mathbf{F}$ and $\mathbf{G}$ column-wisely or element-wisely to gain faster convergence. In ONMF, however, it is difficult to incorporate the orthogonal constraint into column-wise or element-wise coordinate descent updates.

In this paper, we propose a Fast Hierarchical Alternating Least Squares (HALS) algorithm for ONMF (HALS-ONMF). Our algorithm is based on a column-wise update algorithm for NMF proposed by Cichocki and Anh-Huy (2009). To enable such a column-wise update even in ONMF, we derive a set of column-wise orthogonal constraints, taking into consideration both nonnegativity and orthogonality at the same time. Furthermore, we show that the column-wise orthogonal constraint can also be applied to column-wise update algorithms called scalar Block Coordinate Descent for solving Bregman divergence NMF (sBCD-NMF) (Li et al. 2012) where the Frobenius norm in (1) is replaced with more general Bregman divergence (Li et al. 2012). This sBCD-ONMF algorithm is the first algorithm to solve ONMF with Bregman divergence.

The rest of this paper is organized as follows. We summarize previously proposed NMF algorithms and ONMF algorithms by connecting them to the corresponding optimization criteria in Sect. 2. Then we explain HALS-NMF proposed by Cichocki and Anh-Huy (2009) and propose HALS-ONMF with a newly invented column-wise orthogonal constraint in Sect. 3. In Sect. 4, we incorporate the column-wise orthogonal constraint into sBCD-NMF proposed by Li et al. (2012) in order to propose sBCD-ONMF algorithm. In Sect. 5, we present the results of experiments using the conventional and proposed algorithms on several real-life datasets. The conclusion is given in Sect. 6.

We will use a bold uppercase letter for a matrix, such as $\mathbf{X}$, and an italic lowercase letter for a vector such as $\boldsymbol{x}$. Both $\mathbf{X}_{ij}$ and $\boldsymbol{x}_{ij}$ stand for the (i, j)th element in a matrix $\mathbf{X}$. A vector $\mathbf{1}_J \in \mathbb{R}^J$ shows the vector whose elements are of one's. In this paper, NMF means Frobenius norm NMF, unless stated otherwise.

2 Related work

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

2.1 Nonnegative matrix factorization

NMF aims to find a nonnegative matrix $\mathbf{F} = [\boldsymbol{f}_1, \boldsymbol{f}_2, \ldots, \boldsymbol{f}_J] \in \mathbb{R}_+^{N \times J}$ and another nonnegative matrix $\mathbf{G} = [\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_J] \in \mathbb{R}_+^{M \times J}$ whose product approximates a given nonnegative matrix $\mathbf{X} \in \mathbb{R}_+^{N \times M}$:

$$\begin{aligned} &\min_{\mathbf{F},\mathbf{G}} \|\mathbf{X} - \mathbf{F}\mathbf{G}^T\|_F^2, \\ &\text{subject to } \mathbf{F} \geq 0,\ \mathbf{G} \geq 0. \end{aligned} \tag{3}$$

Since the NMF problem is not convex both in $\mathbf{F}$ and $\mathbf{G}$, various iterative algorithms have been proposed (Lee and Seung 2000; Cichocki et al. 2009; Kim and Park 2011; Hsieh and Dhillon 2011). They are categorized according to the unit of updates as follows.

2.1.1 Matrix-wise update algorithms

Lee and Seung (2000) proposed a Multiplicative Update (MU) algorithm. This MU algorithm is one of the efficient algorithms for NMF proposed in the early stage, and thus many extensions followed (e.g., Cai et al. 2011; Cichocki et al. 2009). However, from the viewpoint of convergence, they were not sufficient (Kim et al. 2014). Lin (2007) proposed a Project Gradient Descent (PGD) algorithm for NMF. This algorithm solves an NMF problem by solving Nonnegative Least Squares (NLS) problems for $\mathbf{F}$ and $\mathbf{G}$ alternatively and, gains faster convergence than MU algorithms. The difference in these algorithms is that the MU algorithm uses a fixed step size in the gradient descent, while PGD uses a flexible step size.

2.1.2 Vector-wise update algorithms

Cichocki and Anh-Huy (2009) proposed a Hierarchical Alternating Least Squares (HALS) algorithm. The HALS algorithm solves a set of column-wise NLS problems for each column and updates $\mathbf{F}$ and $\mathbf{G}$ column-wisely. Since column-wise NLS problems can be solved at a high accuracy and efficiency, HALS converges very fast. Kim and Park (2011) proposed an active-set like algorithm that also decomposes a matrix NLS problem into a set of column-wise sub-problems. The difference between HALS and the active-set like method lies in the way to solve a column-wise sub-problem. The former uses the gradient to solve a sub-problem, while the latter uses an active-set method to solve a sub-problem. The active-set method consists of two stages: first, it finds a feasible point in standard NMF, as a nonnegative point, and then it solves a column-wise NLS problem while maintaining feasibility. Li et al. (2012) recently proposed scalar Block Coordinate Descent (sBCD) algorithm. The sBCD algorithm is applicable to not only NMF with Frobenius norm but also NMF with more general Bregman divergence. They used Taylor series expansion to derive the element-wise problem. Since the sBCD algorithm uses the column-wise residual in their update rule, its complexity is the same as that of column-wise update algorithms. Therefore, in this paper, we consider sBCD as a column-wise update algorithm (see Sect. 4.1). All of these vector-wise update algorithms can be regarded as state-of-the-art algorithms, because they converge empirically faster than matrix-wise update algorithms. However, addition of matrix-based constraints such as $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ is still challenging in such column-wise updates.

2.1.3 Element-wise update algorithms

Hsieh and Dhillon (2011) proposed an element-wise update algorithm called a Greedy Coordinate Descent (GCD) algorithm. To the authors' knowledge, it is the fastest algorithm for NMF. The GCD algorithm takes a greedy strategy to decrease the value of the objective function. It selects and updates the most contributable variables for minimization. The low computational cost of GCD is due to the fact that it does not update unnecessary elements. Unfortunately, the GCD algorithm cannot work with such a constraint that affects all elements of one column at the same time. One example of such a constraint is the graph regularized constraint that appears when we minimize $\alpha(\mathrm{tr}(\mathbf{F}^T\mathbf{L}\mathbf{F}))$, where $\mathbf{L}$ is a graph Laplacian matrix of $\mathbf{X}^T\mathbf{X}$. The GCD relies on the fact that, with a fixed $\mathbf{G}$, updating of an element f_{ij} of $\mathbf{F}$ changes only the gradients of elements in the same row $\boldsymbol{f}_{i:}$ because the gradient in $\mathbf{F}$ is given by $(-2\mathbf{X}\mathbf{G} + 2\mathbf{F}\mathbf{G}^T\mathbf{G})$. In more detail, GCD iteratively selects and updates the most contributable variable f_{ij} in the ith row. Unfortunately, the GCD is not applicable to ONMF because the orthogonal condition requires an interaction between different rows.

2.2 Orthogonal NMF

An additional orthogonal constraint, $\mathbf{F}^T\mathbf{F} = \mathbf{I}$, is imposed in ONMF. At first, we briefly review the first ONMF algorithm proposed by Ding et al. (2006) and reveal the problem behind ONMF.

The goal of ONMF is to find a nonnegative orthogonal matrix $\mathbf{F}$ and a nonnegative matrix $\mathbf{G}$ minimizing the following objective function with a Lagrangian multiplier λ,

$$L(\mathbf{F}, \mathbf{G}) = \|\mathbf{X} - \mathbf{F}\mathbf{G}^T\|_F^2 + \mathrm{Tr}[\lambda(\mathbf{F}^T\mathbf{F} - \mathbf{I})]. \tag{4}$$

The KKT complementary condition gives[1]

$$(-2\mathbf{X}\mathbf{G} + 2\mathbf{F}\mathbf{G}^T\mathbf{G} + 2\mathbf{F}\lambda)_{nj}\mathbf{F}_{nj}^2 = 0,\ n = 1, 2, \ldots, N,\ j = 1, 2, \ldots J. \tag{5}$$

Then the update rule of the constrained matrix $\mathbf{F}$ is derived as

$$\mathbf{F}_{nj} \leftarrow \mathbf{F}_{nj}\sqrt{\frac{(\mathbf{X}\mathbf{G})_{nj}}{[\mathbf{F}(\mathbf{G}^T\mathbf{G} + \lambda)]_{nj}}}. \tag{6}$$

The point is how to determine the value of the Lagrange multiplier λ. Since it is not easy to solve this problem for every value of λ, Ding et al. (2006) ignored the nonnegativity and relied only on $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ to have approximate values of off-diagonal elements. By multiplying $\mathbf{F}^T$ from the left in (5), we have

$$\lambda = \mathbf{F}^T\mathbf{X}\mathbf{G} - \mathbf{G}^T\mathbf{G}.$$

Thus, inserting this λ into (6), we have the final update form as

$$\mathbf{F}_{nj} \leftarrow \mathbf{F}_{nj}\sqrt{\frac{(\mathbf{X}\mathbf{G})_{nj}}{(\mathbf{F}\mathbf{F}^T\mathbf{X}\mathbf{G})_{nj}}}.$$

Note that their formulation with the specific values of λ do not strictly satisfy the orthogonality. Nevertheless, this specification is useful in avoiding the *zero-lock* problem appearing both in ONMF and in NMF: Once an element becomes zero in the middle of iterations, the element will not be recasted in the following steps [see the multiplicative update rule (6)].

[1] Hereafter, we will not state the nonnegative constraint explicitly.

Besides, when the orthogonality is strictly posed with nonnegativity, each row vector of $\mathbf{F}$ must have only one non-zero value. That is, any algorithm using a multiplicative update rule falls easily into a hole of the zero-lock problem. Therefore, ONMF algorithms put the first priority on the approximation while loosening the degree of orthogonality.

As a result of compromise, an ONMF algorithm can be seen as an algorithm that balances the trade-off between orthogonality and approximation with a weighting parameter λ as seen in (2). We do not categorize ONMF algorithms by the unit of updates because all conventional ONMF algorithms are based only on matrix-wise updates. Rather, those algorithms should be categorized according to whether the algorithm employs a weighting parameter or not. If an algorithm minimizes an objective function with a weighting parameter and if the value is not appropriately chosen, then the algorithm would fail in either acceptable degree of approximation or orthogonality. Such a failure has often been reported in past experimental results (Li et al. 2010; Mirzal 2014; Pompili et al. 2012).

2.2.1 Without a weighting parameter

Ding et al. (2006) proposed the first ONMF algorithm based on the MU algorithm (Lee and Seung 2000). This algorithm has no weighting parameter. It solves approximately the Lagrangian (4) as we reviewed. Yoo and Choi (2008) also proposed an MU-based algorithm. They used the gradient on the Stiefel manifold that is the set of all orthogonal matrices. The gradient on the Stiefel manifold is compatible with that of the MU algorithm because the manifold constrains every matrix to be orthogonal and the employed MU algorithm guarantees nonnegative values.[2]

2.2.2 With a weighting parameter

Mirzal proposed a convergent algorithm that is also based on the MU algorithm in practice. He proposed two algorithms in Mirzal (2014), one of which is the same as the one by Li et al. (2010). The first algorithm introduced a weighting parameter α instead of the Lagrangian multiplier λ in (4). The second algorithm was a convergent algorithm. The convergence of the algorithm is proved, but the computational cost is high. In this algorithm, the zero-lock problem was forcibly avoided by replacing zero values with a small positive value ϵ. There are algorithms that put the first priority on nonnegativity rather than orthogonality. Pompili et al. (2012) tackled directly the zero-lock problem. They employed the Augmented Lagrangian method. In more detail, they used the gradient on the Stiefel manifold and explicitly introduced a Lagrangian multiplier ψ for nonnegativity. The initial value of the Lagrangian was approximated to a smaller value in order to avoid the zero-lock problem. They increase the value of ψ gradually to strengthen the nonnegativity while the iteration is repeated. As a result, the nonnegativity was not strictly guaranteed in the algorithm. In addition, it has three parameters to be set appropriately for orthogonality, nonnegativity and step size.

There are mainly two problems to be solved in order to develop fast ONMF algorithms. First, we have to incorporate the matrix-type orthogonal condition $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ into column-wise or element-wise updating NMF algorithms. This is necessary to obtain efficiency. Next, we need to solve the zero-lock problem. This is necessary to find an appropriate balance between orthogonality and nonnegativity without a weighting parameter. This problem prevents us

[2] In general, the resultant constrained matrix by ONMF in Yoo and Choi (2008) also does not satisfy strict orthogonality because the MU algorithm is gradient descent with a fixed step size, and thus, it may undershoot or overshoot.

Table 1 A summary of categorization of ONMF algorithms. Frobenius, KL and Bregman denotes distortions used for measuring the degree of approximation

Author (year)	Updates		Weighting (YES/NO)	Distortion measure
	Matrix	Vector		
Ding et al. (2006)	MU		NO	Frobenius
Yoo and Choi (2008)	MU		NO	Frobenius
Li et al. (2010)	MU		YES	Frobenius/KL
Pompili et al. (2012)	PGD		YES	Frobenius
Mirzal (2014)	MU		YES	Frobenius
Proposed (HALS-ONMF)		HALS	NO	Frobenius
Proposed (sBCD-ONMF)		sBCD	NO	Bregman

from using the Lagrangian and alternatively forces us to take a balance between orthogonality and nonnegativity appropriately. In this paper, we show a way to realize two things in ONMF algorithms (Table 1).

3 Hierarchical alternating least squares algorithm for ONMF

In this section, we show a way of utilizing the HALS for ONMF. First, we briefly review the HALS for standard NMF and then describe how to incorporate the orthogonal constraint column-wisely to propose HALS-ONMF.

3.1 Hierarchical alternating least squares for NMF

The key idea of HALS is efficient decomposition of the residual. Suppose that all of the elements of matrices $\mathbf{F}$ and $\mathbf{G}$ are fixed except for the jth columns $\boldsymbol{f}_j$ and $\boldsymbol{g}_j$. Since $\mathbf{F}\mathbf{G}^T = \sum_{j=1}^{J} \boldsymbol{f}_j \boldsymbol{g}_j^T$, the objective function (3) can be minimized by finding more appropriate $\boldsymbol{f}_j$ and $\boldsymbol{g}_j$ than the current ones such as

$$\min_{\boldsymbol{f}_j, \boldsymbol{g}_j} J_j = \left\| \mathbf{X}^{(j)} - \boldsymbol{f}_j \boldsymbol{g}_j^T \right\|_F^2, \tag{7}$$

where $\mathbf{X}^{(j)} \triangleq \mathbf{X} - \sum_{k \neq j} \boldsymbol{f}_k \boldsymbol{g}_k^T$ is a residue. Since $\boldsymbol{f}_j$ affects only $\boldsymbol{g}_j$, HALS alternatively minimizes (7) for $j = 1, 2, \ldots, J, 1, 2, \ldots$, until convergence, keeping the nonnegative constraints, $\boldsymbol{f}_j \geq 0$ and $\boldsymbol{g}_j \geq 0$. This objective function (7) with nonnegative constraints can be considered as an Nonnegative Least Squares (NLS) problem. HALS solves the set of such NLS problems.

In order to find a stationary point, the gradients of (7) in $\boldsymbol{f}_j$ and $\boldsymbol{g}_j$ are calculated:

$$\mathbf{0} = \frac{\partial J_j}{\partial \boldsymbol{f}_j} = \boldsymbol{f}_j \boldsymbol{g}_j^T \boldsymbol{g}_j - \mathbf{X}^{(j)} \boldsymbol{g}_j, \text{ and} \tag{8}$$

$$\mathbf{0} = \frac{\partial J_j}{\partial \boldsymbol{g}_j} = \boldsymbol{g}_j \boldsymbol{f}_j^T \boldsymbol{f}_j - \mathbf{X}^{(j)T} \boldsymbol{f}_j. \tag{9}$$

Hence, we have the following update rules:

$$\boldsymbol{f_j} \leftarrow \frac{1}{\boldsymbol{g_j}^T \boldsymbol{g}_j}[\mathbf{X}^{(j)} \boldsymbol{g}_j]_+, \tag{10}$$

$$\boldsymbol{g_j} \leftarrow \frac{1}{\boldsymbol{f_j}^T \boldsymbol{f}_j}[\mathbf{X}^{(j)T} \boldsymbol{f}_j]_+, \tag{11}$$

where $[x]_+ = \max(\epsilon, x)$ (ϵ being a sufficiently small positive value).

Without loss of generality, we may normalize so as to $\|\boldsymbol{f}_j\|_2^2 = 1$ after updating. Assuming this normalization, we can remove $\boldsymbol{g}_j^T \boldsymbol{g}_j$ and $\boldsymbol{f}_j^T \boldsymbol{f}_j$ from (10) and (11), respectively. Now the update rules (10) and (11) become simpler:

$$\begin{aligned} \boldsymbol{f_j} &\leftarrow [\mathbf{X}^{(j)} \boldsymbol{g}_j]_+, \quad \text{and, after re-normalization to } \|\boldsymbol{f}_j\|_2^2 = 1, \\ \boldsymbol{g_j} &\leftarrow [\mathbf{X}^{(j)T} \boldsymbol{f}_j]_+. \end{aligned}$$

Since $\mathbf{X}^{(j)} = \mathbf{X} - \sum_{k \neq j} \boldsymbol{f}_k \boldsymbol{g}_k^T = \mathbf{X} - \mathbf{F}\mathbf{G}^T + \boldsymbol{f}_j \boldsymbol{g}_j^T$, we finally obtain the following column-wise update rules:

$$\begin{aligned} \boldsymbol{f_j} &\leftarrow \left[(\mathbf{XG})_j - \mathbf{F}(\mathbf{G}^T\mathbf{G})_j + \boldsymbol{f}_j \boldsymbol{g}_j^T \boldsymbol{g}_j \right]_+, \text{ and} \\ \boldsymbol{g_j} &\leftarrow \left[(\mathbf{X}^T\mathbf{F})_j - \mathbf{G}(\mathbf{F}^T\mathbf{F})_j + \boldsymbol{g}_j \boldsymbol{f}_j^T \boldsymbol{f}_j \right]_+. \end{aligned}$$

Note that $\mathbf{XG}$ and $\mathbf{G}^T\mathbf{G}$ do not change their values while vectors $\boldsymbol{f}_j$ ($j = 1, \ldots, J$) are updated. Therefore, HALS computes $\mathbf{XG}$ and $\mathbf{G}^T\mathbf{G}$ before updating those vectors. Similarly, we per-calculate $\mathbf{X}^T\mathbf{F}$ and $\mathbf{F}^T\mathbf{F}$ before updating $\boldsymbol{g}_j$ ($j = 1, \ldots, J$).[3] This is the HALS algorithm usable for regular NMF.

3.2 Column-wise orthogonal constraint

Since $\boldsymbol{f}_j$ affects the other columns in $\mathbf{F}^T\mathbf{F}$, the orthogonal constraint cannot be directly introduced into the HALS algorithm above. In this paper, we exploit a simple fact that if the sum of nonnegative values is zero, then all of the values are zero. Since the orthogonal condition $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ means $\boldsymbol{f}_k^T \boldsymbol{f}_j = 0$ for every $k \neq j$, we can use a single condition $\sum_{k \neq j} \boldsymbol{f}_k^T \boldsymbol{f}_j = 0$ for fixed j coupled with $\boldsymbol{f}_k^T \boldsymbol{f}_j \geq 0$, instead of $J-1$ conditions $\boldsymbol{f}_k^T \boldsymbol{f}_j = 0$ for every k ($\neq j$). That is, one matrix condition $\mathbf{F}^T\mathbf{F} = \mathbf{I}$ is equivalently replaced with $2J$ column-wise constraints of $\boldsymbol{f}_j^T \boldsymbol{f}_j = 1$ and $\sum_{k \neq j} \boldsymbol{f}_k^T \boldsymbol{f}_j = 0$ for every j. As will be shown, the newly derived column-wise constraints can be updated with $O(M)$ for each column (M being the number of rows of $\mathbf{X}$ to be factorized).

Now it suffices to impose the conditions

$$\mathbf{F}^{(j)T} \boldsymbol{f}_j \triangleq \sum_{k \neq j} \boldsymbol{f}_k^T \boldsymbol{f}_j = 0, \quad j = 1, 2, \ldots, J. \tag{12}$$

In addition, we normalize each column vector so as to $\|\boldsymbol{f}_j\|^2 = \boldsymbol{f}_j^T \boldsymbol{f}_j = 1$ to satisfy $\mathbf{F}^T\mathbf{F} = \mathbf{I}$. Thus, we introduce constraint $\mathbf{F}^{(j)T} \boldsymbol{f}_j = 0$ ($j = 1, 2, \ldots, J$) into (4) as the column-wise orthogonal constraint. The nonnegativity of the elements is preserved with the ϵ-truncate function $[\,]_+$.

[3] Sometimes it is called Fast HALS.

3.3 HALS-ONMF

With the derived column-wise constraint (12), the localized objective function is formulated as a Lagrangian:

$$L(\boldsymbol{f}_j, \boldsymbol{g}_j, \lambda_j) = \left\| \mathbf{X}^{(j)} - \boldsymbol{f}_j \boldsymbol{g}_j^T \right\|_F^2 + \lambda_j \left(\mathbf{F}^{(j)T} \boldsymbol{f}_j \right), \quad \text{where}$$
$$\mathbf{X}^{(j)} = \mathbf{X} - \sum_{k \neq j} \boldsymbol{f}_k \boldsymbol{g}_k^T,$$
$$\mathbf{F}^{(j)} = \sum_{k \neq j} \boldsymbol{f}_k, \quad \lambda_j \geq 0.$$

The gradient is given as

$$\frac{\partial L}{\partial \boldsymbol{f}_j} = -2\mathbf{X}^{(j)} \boldsymbol{g}_j + 2\boldsymbol{f}_j \boldsymbol{g}_j^T \boldsymbol{g}_j + \lambda_j \mathbf{F}^{(j)}. \tag{13}$$

By solving $\partial L / \partial \boldsymbol{f}_j = \mathbf{0}$ and forcibly keeping the nonnegativity, we obtain the update rule, under the assumption of normalization of $\boldsymbol{f}_j^T \boldsymbol{f}_j = 1$, as post-processing:

$$\boldsymbol{f}_j \leftarrow \left[\mathbf{X}^{(j)} \boldsymbol{g}_j - \frac{\lambda_j}{2} \mathbf{F}^{(j)} \right]_+. \tag{14}$$

Unfortunately, the setting of the value of λ still remains as a problem. In this study, we take the same way as Ding et al. did in (2006). By multiplying $\mathbf{F}^{(j)}$ from the left in (13) and using $\mathbf{F}^{(j)T} \boldsymbol{f}_j = \mathbf{0}$, we obtain

$$\lambda_j = \frac{2\mathbf{F}^{(j)T} \mathbf{X}^{(j)} \boldsymbol{g}_j}{\mathbf{F}^{(j)T} \mathbf{F}^{(j)}}.$$

Hence, (14) becomes

$$\boldsymbol{f}_j \leftarrow \left[\mathbf{X}^{(j)} \boldsymbol{g}_j - \frac{\mathbf{F}^{(j)T} \mathbf{X}^{(j)} \boldsymbol{g}_j}{\mathbf{F}^{(j)T} \mathbf{F}^{(j)}} \mathbf{F}^{(j)} \right]_+. \tag{15}$$

Since the orthogonal constraint $\mathbf{F}^{(j)T} \boldsymbol{f}_j = 0$ does not affect $\boldsymbol{g}_j$, we can use the same update rule as HALS-NMF, that is, with (11),

$$\boldsymbol{f}_j \leftarrow \left[\mathbf{X}^{(j)} \boldsymbol{g}_j - \frac{\mathbf{F}^{(j)T} \mathbf{X}^{(j)} \boldsymbol{g}_j}{\mathbf{F}^{(j)T} \mathbf{F}^{(j)}} \mathbf{F}^{(j)} \right]_+, \quad \text{and}$$
$$\boldsymbol{g}_j \leftarrow \left[\mathbf{X}^{(j)T} \boldsymbol{f}_j \right]_+.$$

Using $\mathbf{X}^{(j)} = \mathbf{X} - \sum_{k \neq j} \boldsymbol{f}_k \boldsymbol{g}_k^T = \mathbf{X} - \mathbf{F}\mathbf{G}^T + \boldsymbol{f}_j \boldsymbol{g}_j^T$, we have the final form of updating rules:

$$\boldsymbol{f}_j \leftarrow \left[\boldsymbol{h} - \frac{\mathbf{F}^{(j)T} \boldsymbol{h}}{\mathbf{F}^{(j)T} \mathbf{F}^{(j)}} \mathbf{F}^{(j)} \right]_+,$$
$$\boldsymbol{f}_j \leftarrow \boldsymbol{f}_j / \|\boldsymbol{f}_j\|_2^2, \quad \text{and}$$
$$\boldsymbol{g}_j \leftarrow \left[(\mathbf{X}^T \mathbf{F})_j - \mathbf{G}(\mathbf{F}^T \mathbf{F})_j + \boldsymbol{g}_j \boldsymbol{f}_j^T \boldsymbol{f}_j \right]_+, \quad \text{where}$$
$$\boldsymbol{h} = (\mathbf{X}\mathbf{G})_j - \mathbf{F}(\mathbf{G}^T \mathbf{G})_j + \boldsymbol{f}_j \boldsymbol{g}_j^T \boldsymbol{g}_j.$$

Algorithm 1 Fast HALS-Orthogonal NMF

Input: Nonnegative matrix $\mathbf{X}$, Number of components J
Output: Decomposing nonnegative matrices $\mathbf{F}$ and $\mathbf{G}$ such that $\mathbf{X} \simeq \mathbf{F}\mathbf{G}^T$ and $\mathbf{F}^T\mathbf{F} \cong \mathbf{I}$.

Initialize $\mathbf{F}$ and $\mathbf{G}$ arbitrary.
$\mathbf{U} = \mathbf{F}\mathbf{1}_J$
repeat
  $\mathbf{A} = \mathbf{X}\mathbf{G}$
  $\mathbf{B} = \mathbf{G}^T\mathbf{G}$
  for $j = 1$ **to** J **do**
    $\mathbf{F}^{(j)} = \mathbf{U} - \boldsymbol{f}_j$
    $\boldsymbol{h} = \mathbf{A}_j - \mathbf{F}\mathbf{B}_j + \mathbf{B}_{jj}\boldsymbol{f}_j$
    $\boldsymbol{f}_j = [\boldsymbol{h} - \frac{\mathbf{F}^{(j)}\boldsymbol{h}}{\mathbf{F}^{(j)T}\mathbf{F}^{(j)}}\mathbf{F}^{(j)}]_+$
    $\boldsymbol{f}_j = \boldsymbol{f}_j / \parallel \boldsymbol{f}_j \parallel^2$
    $\mathbf{U} = \mathbf{F}^{(j)} + \boldsymbol{f}_j$
  end for
  $\mathbf{C} = \mathbf{X}^T\mathbf{F}$
  $\mathbf{D} = \mathbf{F}^T\mathbf{F}$
  for $j = 1$ **to** J **do**
    $\boldsymbol{g}_j \leftarrow [\mathbf{C}_j - \mathbf{G}\mathbf{D}_j + \mathbf{D}_{jj}\boldsymbol{g}_j]_+$
  end for
until Convergence criterion is satisfied.

The zero-lock problem is resolved by $[\ ]_+$ operation as it is in Mirzal (2014). The proposed HALS-ONMF algorithm is shown in Algorithm 1.

This vector-wise update algorithm is faster than conventional matrix-wise update algorithms for the following reason. The matrix-wise update rule (6) is derived from (5), while the vector-wise update rule (15) of the proposed HALS-ONMF is derived from (13). The former comes from the KKT complementary condition which is just a necessary condition for the solution to minimize (4). Therefore, there is no guarantee for the updating to be optimum in each iteration. While, in the latter, the corresponding optimization problem can be solved analytically in a closed form. Therefore, the updating is always optimal in each iteration.

4 ONMF with Bregman divergence

In this section, we consider a wider class of ONMF problems; that is, Bregman divergence is introduced instead of the Frobenius norm to measure the degree of approximation. In the case of NMF, Li et al. (2012) already proposed a column-wise update algorithm called scalar Block Coordinate Descent (sBCD) to solve Bregman divergence NMF. In this paper, we develop Bregman divergence ONMF, by incorporating the column-wise orthogonal constraint into their sBCD algorithm. We first briefly review the sBCD-algorithm (Li et al. 2012) and then explain how our column-wise orthogonal constraint can be incorporated in sBCD.

4.1 Scalar block coordinate descent algorithm (sBCD)

The objective function is now given as

$$\begin{aligned} &\min_{\mathbf{F},\mathbf{G}} D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T), \\ &\text{subject to } \mathbf{F} \geq 0, \mathbf{G} \geq 0, \end{aligned} \tag{16}$$

Table 2 Examples of Bregman Divergence

Description	Function $\phi(a)$	$D_\phi(a\|\|b)$
Frobenius norm	$\frac{a^2}{2}$	$(a-b)^2/2$
KL-divergence	$a\log a$	$a\log\frac{a}{b}-a+b$
IS-divergence	$-\log a$	$\frac{a}{b}-\log\frac{a}{b}$
β-Divergence	$\frac{a^{\beta+1}-(\beta+1)a+\beta}{\beta(\beta+1)}$	$\frac{1}{\beta(\beta+1)}(a^{\beta+1}-b^{\beta+1}-(\beta+1)b^\beta(a-b))$

where $D_\phi(\mathbf{A}||\mathbf{B})$ is a Bregman divergence between matrices $\mathbf{A}$ and $\mathbf{B}$ using a strictly convex function ϕ. The definition of Bregman divergence is as follows.

Definition 1 (*Bregman divergence*) Let $\phi : S \subseteq \mathbb{R} \to \mathbb{R}$ be a strictly convex function with the continuous first derivation $\nabla\phi$. Then, Bregman divergence corresponding to ϕ, $D_\phi : S \times \mathrm{int}(S) \to \mathbb{R}_+$, is defined as $D_\phi(x, y) = \phi(x) - \phi(y) - \nabla\phi(y)(x-y)$. Here $\mathrm{int}(S)$ is the interior of S.

A Bregman divergence for scalars is extended to the one for matrices by $D_\phi(\mathbf{A}||\mathbf{B}) = \sum_{m,n} D_\phi(\mathbf{A}_{mn}|\mathbf{B}_{mn})$. Bregman divergences include many well-known divergences such as Frobenius norm and KL-divergence (Table 2). Recently, Li et al. proposed a column-wise update algorithm for Bregman divergence NMF. [4] The key idea of the update rules is Taylor series of Bregman divergences.

Let $E_t(a||b) \triangleq |a-b|^t$ and $E_t(\mathbf{X}||\mathbf{X}') \triangleq \sum_{mn} |x_{mn} - x'_{mn}|^t$ be the tth power of t-norm distance. Then, for $\mathbf{X}' = \mathbf{F}\mathbf{G}^T$, we have

$$\min_{\mathbf{X}'} E_t(\mathbf{X}||\mathbf{X}') = \min_{\mathbf{F}\mathbf{G}} E_t(\mathbf{X}||\mathbf{F}\mathbf{G}^T) = \min_{\forall j \boldsymbol{f}_j, \boldsymbol{g}_j} E_t\left(\mathbf{X}^{(j)}||\boldsymbol{f}_j\boldsymbol{g}_j^T\right).$$

We want to connect $E_t(\mathbf{X}^{(j)}||\boldsymbol{f}_j\boldsymbol{g}_j^T)$ with Bregman divergence $D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T)$ to minimize $E_t(\mathbf{X}^{(j)}||\boldsymbol{f}_j\boldsymbol{g}_j^T)$. In the scalar case, by applying the Taylor series of $\phi(x)$ at $x = b$ to $\phi(a)$, we have

$$\begin{aligned} D_\phi(a||b) &= \phi(a) - \phi(b) - \nabla\phi(b)(a-b) \\ &= \nabla\phi(b)(a-b) + \sum_{t=2}^{\infty} \frac{\nabla^t\phi(b)}{t!}(a-b)^t \\ &\quad - \nabla\phi(b)(a-b) \quad \text{(Taylor series of the first two terms)} \\ &= \sum_{t=2}^{\infty} \frac{\nabla^t\phi(b)}{t!}(a-b)^t \\ &= \sum_{t=2}^{\infty} \frac{\nabla^t\phi(b)}{t!}(-\mathrm{sgn}(b-a))^t E_t(a||b), \end{aligned} \tag{17}$$

where $\nabla^t\phi(b)$ is the t-order derivative of $\phi(x)$ at $x = b$. The last equation comes from the relation: $(a-b)^t = (\mathrm{sgn}(a-b))^t|a-b|^t$. Hence, as a natural extension, $D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T)$ can be re-written as

[4] They proposed the scalar Block Coordinate Descent algorithm as an element-wise update algorithm. However, their update rules need to re-calculate the residual column-wisely. Therefore, we consider their algorithm as a column-wise update algorithm.

$$D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T) = \sum_{mn}\sum_{t=2}^{\infty} \frac{\nabla^t\phi\left(x'_{mn}\right)}{t!}\left(-\mathrm{sgn}\left(x'_{mn}-x_{mn}\right)\right)^t E_t\left(x^{(j)}_{mn}||f_{mj}g_{nj}\right),$$

where $x^{(j)}_{mn} = (\mathbf{X} - \sum_{k\neq j}\boldsymbol{f}_k\boldsymbol{g}_k)_{mn}$ and $x'_{mn} = (\mathbf{F}\mathbf{G}^T)_{mn}$. Thus, we can use the partial derivation of $E_t(x^{(j)}_{mn}||f_{mj}g_{nj})$ instead of that of $D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T)$. Since

$$\begin{aligned}&\frac{\partial}{\partial f_{mj}}\left(\frac{\nabla^t\phi\left(x'_{mn}\right)}{t!}\left(-\mathrm{sgn}\left(f_{mj}g_{nj}-x^{(j)}_{mn}\right)\right)^t E_t\left(x^{(j)}_{mn}||f_{mj}g_{nj}\right)\right)\\ &\quad= -g_{nj}\frac{\nabla^t\phi\left(x'_{mn}\right)}{t-1!}\left(x^{(j)}_{mn}-f_{mj}g_{nj}\right)^{t-1} + g_{nj}\frac{\nabla^{t+1}\phi\left(x'_{mn}\right)}{t!}\left(x^{(j)}_{mn}-f_{mj}g_{nj}\right)^t,\end{aligned}$$

with (17), we have

$$\begin{aligned}\frac{\partial D_\phi\left(x_{mn}||x'_{mn}\right)}{f_{mj}} &= g_{nj}\nabla^2\phi\left(x'_{mn}\right)\left(f_{mj}g_{nj}-x^{(j)}_{mn}\right)\\ &\quad+\sum_{t=2}^{\infty}\left(-g_{nj}\frac{\nabla^{t+1}\phi\left(x'_{mn}\right)}{t!}\left(x^{(j)}_{mn}-f_{mj}g_{nj}\right)^t\right.\\ &\quad\left.+\,g_{nj}\frac{\nabla^t\phi\left(x'_{mn}\right)}{t!}\left(x^{(j)}_{mn}-f_{mj}g_{nj}\right)^t\right)\\ &= g_{nj}\nabla^2\phi\left(x'_{mn}\right)\left(f_{mj}g_{nj}-x^{(j)}_{mn}\right).\end{aligned}$$

Taking the sum over the rows and columns, we obtain the gradient of $D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T)$ in f_{nj}:

$$\frac{\partial D_\phi\left(\mathbf{X}||\mathbf{F}\mathbf{G}^T\right)}{\partial f_{nj}} = \sum_{n=1}^{N} g_{nj}\nabla^2\phi\left(x'_{mn}\right)\left(f_{mj}g_{nj}-x^{(j)}_{mn}\right). \tag{18}$$

Finally, the update rule of sBCD is given by

$$f_{nj} \leftarrow \left[\frac{\sum_{n=1}^{N}\nabla^2\phi\left(x'_{mn}\right)x^{(j)}_{mn}g_{nj}}{\sum_{n=1}^{N}\nabla^2\phi\left(x'_{mn}\right)g_{nj}g_{nj}}\right]_+. \tag{19}$$

This sBCD algorithm (19) needs to calculate column-wise residual $\mathbf{X}^{(j)} = \mathbf{X} - \sum_{k\neq j}\boldsymbol{f}_k\boldsymbol{g}_k$ for $x^{(j)}_{mn}$ in (19). Therefore, instead of the element-wise update (19), we adopt the following:

$$\boldsymbol{f}_j \leftarrow \left[\frac{(\nabla^2\phi(\mathbf{F}\mathbf{G}^T)\odot\mathbf{X}^{(j)})\boldsymbol{g}_j}{\nabla^2\phi(\mathbf{F}\mathbf{G}^T)\boldsymbol{g}_j^2}\right]_+. \tag{20}$$

4.2 Bregman divergence ONMF

Now, we introduce the orthogonal constraint into Bregman divergence NMF to have Bregman divergence ONMF. The minimization problem of Bregman divergence ONMF is given by

$$\begin{aligned}&\min_{\mathbf{F},\mathbf{G}} D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T)\\ &\text{subject to } \mathbf{F}\geq 0, \mathbf{G}\geq 0,\ \mathbf{F}^T\mathbf{F}=\mathbf{I}.\end{aligned} \tag{21}$$

For the same reason as that stated before, we solve its relaxed version:

$$\begin{aligned}&\min_{\mathbf{F},\mathbf{G}} D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T) + \lambda\|\mathbf{F}^T\mathbf{F} - \mathbf{I}\| \\ &\text{subject to } \mathbf{F} \geq 0, \mathbf{G} \geq 0.\end{aligned}$$

This problem can be re-written equivalently and column-wisely as

$$\begin{aligned}O = \min_{\forall j\ \boldsymbol{f}_j, \boldsymbol{g}_j, \lambda_j} D_\phi\left(\mathbf{X}||\left(\sum_{k\neq j}\boldsymbol{f}_k\boldsymbol{g}_k^T\right) + \boldsymbol{f}_j\boldsymbol{g}_j^T\right) + \lambda\mathbf{F}^{(j)T}\boldsymbol{f}_j, \\ \text{where } \mathbf{F}^{(j)} = \sum_{k\neq j}\boldsymbol{f}_k,\ \lambda_j \geq 0.\end{aligned} \tag{22}$$

Note that the first term of RHS of (22) is equivalent to (21). Hence, we have

$$\frac{\partial O}{\partial \boldsymbol{f}_j} = \left(\nabla^2\phi\left(\mathbf{X}'_{mn}\right) \odot \left(\boldsymbol{f}_j\boldsymbol{g}_j^T - \mathbf{X}^{(j)}\right)\right)\boldsymbol{g}_j + \lambda_j\mathbf{F}^{(j)}. \tag{23}$$

To determine the value of the Lagrangian multiplier λ_j, we again assume $\mathbf{F}^{(j)T}\boldsymbol{f}_j = 0$ and multiply $\mathbf{F}^{(j)T}$ from the left to (23) to be zero. This gives

$$\lambda_j\mathbf{F}^{(j)T}\mathbf{F}^{(j)} = -\mathbf{F}^{(j)T}\left(\boldsymbol{f}_j\boldsymbol{g}_j^T \odot \nabla^2\phi(\mathbf{F}\mathbf{G}^T)\right)\boldsymbol{g}_j + \mathbf{F}^{(j)T}\left(\mathbf{X}^{(j)} \odot \nabla^2\phi(\mathbf{F}\mathbf{G}^T)\right)\boldsymbol{g}_j.$$

Under nonnegativity $\boldsymbol{f}_j \geq 0$ and $\mathbf{F}^{(j)} \geq 0$, with the assumption $\mathbf{F}^{(j)T}\boldsymbol{f}_j = 0$, we have

$$\boldsymbol{f}_j, \mathbf{F}^{(j)} \geq 0 \text{ and } \mathbf{F}^{(j)T}\boldsymbol{f}_j = 0 \Rightarrow \mathbf{F}^{(j)T}\left(\boldsymbol{f}_j\boldsymbol{g}_j^T \odot \nabla^2\phi(\mathbf{F}\mathbf{G}^T)\right)\boldsymbol{g}_j = 0.$$

This is because the row indices of zero values of $(\boldsymbol{f}_j\boldsymbol{g}_j^T \odot \nabla^2\phi(\mathbf{F}\mathbf{G}^T))\boldsymbol{g}_j$ are the same as those of $\boldsymbol{f}_j$. Hence, we may set λ_j to

$$\lambda_j = \frac{\mathbf{F}^{(j)T}(\mathbf{X}^{(j)} \odot \nabla^2\phi(\mathbf{F}\mathbf{G}^T))\boldsymbol{g}_j}{\mathbf{F}^{(j)T}\mathbf{F}^{(j)}}.$$

Then the update rule of sBCD-ONMF becomes

$$\boldsymbol{f}_j \leftarrow \left[\frac{(\nabla^2\phi(\mathbf{F}\mathbf{G}^T) \odot \mathbf{X}^{(j)})\boldsymbol{g}_j - \frac{\mathbf{F}^{(j)T}(\mathbf{X}^{(j)} \odot \nabla^2\phi(\mathbf{F}\mathbf{G}^T))\boldsymbol{g}_j}{\mathbf{F}^{(j)T}\mathbf{F}^{(j)}}\mathbf{F}^{(j)}}{\nabla^2\phi(\mathbf{F}\mathbf{G}^T)\boldsymbol{g}_j^2}\right]_+. \tag{24}$$

If we use $\phi(x) = x^2/2$ corresponding to the Frobenius norm, it is easy to verify $\nabla^2\phi(\mathbf{F}\mathbf{G}^T) = \mathbf{1}$. It implies that (24) is equivalent to (15) with post-processing normalization $\|\boldsymbol{f}_j^T\|_2^2 = 1$.

This sBCD-ONMF algorithm is an extension of the previous HALS-ONMF algorithm, but its convergence is slower because sBCD-ONMF needs to update the column-wise residual in addition to the updating of each column, while HALS-ONMF does not need to do so for the residual. The proposed sBCD-ONMF algorithm is shown in Algorithm 2.

4.3 Relation to Bregman hard clustering

The original ONMF is known to be related to k-means clustering (Ding et al. 2006). So, in this section, we make clear the relationship between Bregman divergence ONMF and Bregman Hard Clustering proposed by Banerjee et al. (2005b). The criterion of Bregman

Algorithm 2 sBCD-Orthogonal NMF (a generalized HALS-ONMF)

Input: Nonnegative matrix $\mathbf{X}$, Number of components J and functionϕ for a Bregman Divergence
Output: Decomposing nonnegative matrices $\mathbf{F}$ and $\mathbf{G}$ such that $\mathbf{X} \simeq \mathbf{F}\mathbf{G}^T$ and $\mathbf{F}^T\mathbf{F} \cong \mathbf{I}$.

Initialize $\mathbf{F}$ and $\mathbf{G}$ arbitrary.
$\mathbf{U} = \mathbf{F}\mathbf{1}_J$
$\mathbf{X}' = \mathbf{F}\mathbf{G}^T$
$\mathbf{E} = \mathbf{X} - \mathbf{X}'$
repeat
  $\mathbf{B} = \nabla^2\mathbf{X}'$
  for $j = 1$ **to** J **do**
    $\mathbf{X}^{(j)} = \mathbf{E} + \boldsymbol{f}_j\boldsymbol{g}_j^T$
    $\mathbf{F}^{(j)} = \mathbf{U} - \boldsymbol{f}_j$
    $\boldsymbol{h} = (\mathbf{B} \odot \mathbf{X}^{(j)})\boldsymbol{g}_j$
    $\boldsymbol{f}_j = \left[\frac{\boldsymbol{h} - \frac{\mathbf{F}^{(j)T}\boldsymbol{h}}{\mathbf{F}^{(j)T}\mathbf{F}^{(j)}}\mathbf{F}^{(j)}}{\mathbf{B}\boldsymbol{g}_j^2}\right]_+$
    $\boldsymbol{g}_j = \left[\frac{(\mathbf{B}\odot\mathbf{X}^{(j)})^T\boldsymbol{f}_j}{\mathbf{B}^T\boldsymbol{f}_j^2}\right]_+$
    $\mathbf{U} = \mathbf{F}^{(j)} + \boldsymbol{f}_j$
    $\mathbf{E} = \mathbf{X}^{(j)} - \boldsymbol{f}_j\boldsymbol{g}_j^T$
  end for
  $\mathbf{X}' = \mathbf{F}\mathbf{G}^T$
until Convergence criterion is satisfied.

hard clustering to minimize is a natural extension of that of k-means clustering as shown below:

$$\min_{\pi_{j=1,2,\ldots,J}} \sum_{j=1}^{J} \sum_{n\in\pi_j} D_\phi(\boldsymbol{x}_n||\boldsymbol{\mu}_j), \tag{25}$$

where π_j for $j = 1, 2, \ldots, J$ is a set of disjoint clusters and $\mu_j = \sum_{n\in\pi_j} \frac{1}{|\pi_j|}\boldsymbol{x}_n$ is the centroid of cluster π_j. Then, we have the following theorem.

Theorem 1 (Equivalence between Bregman divergence ONMF and Bregman hard clustering) *The minimization problem of Bregman divergence ONMF* (21) *is equivalent to that of the Bregman hard clustering defined in* (25).

Proof Let us suppose a given data matrix $\mathbf{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_N] \in \mathbb{R}^{M\times N}$ and impose the orthogonal constraint into $\mathbf{G}$ instead of $\mathbf{F}$, that is, $\mathbf{G}^T\mathbf{G} = \mathbf{I}$.[5] We first consider the minimization problem for ONMF with Bregman divergence:

$$D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T) = \sum_{n=1}^{N} D_\phi(\boldsymbol{x}_n||(\mathbf{F}\mathbf{G}^T)_n),$$

where $(\mathbf{F}\mathbf{G}^T)_n$ denotes the nth column vector of matrix $\mathbf{F}\mathbf{G}^T$. As stated before, each row vector of the orthogonal nonnegative matrix $\mathbf{G}$ has only one non-zero value. In a clustering

[5] This formulation is acceptable since the problem is equivalent to problem (1) with transpose $\mathbf{X} \leftarrow \mathbf{X}^T$.

Table 3 Datasets used in the experiments

Dataset	Size	#nnz	Source
20Newsgroup	61,188 × 18,774	2,435,219	Document
TDT	36,771 × 9394	1,224,135	Document
RCV	29,992 × 9625	730,879	Document
Reuters21678	18,993 × 8293	389,455	Document
MNIST	784 × 70,000	10,505,375	Image
ORL64 × 64	4096 × 400	1,638,400	Image

Here #nnz is the number of non-zero values

task, this non-zero value corresponds to the clustering index that the data belong to. Therefore, we can rewrite the minimization problem as

$$\sum_{n=1}^{N}\sum D_\phi(\boldsymbol{x}_n||(\mathbf{F}\mathbf{G}^T)_n) = \sum_{j=1}^{J}\sum_{n:g_{nj}\neq 0} D_\phi(\boldsymbol{x}_n||g_{nj}\boldsymbol{f}_j).$$

According to Ding et al. (2008), let us impose the row normalization condition

$$g_{nj} = 1.$$

Then, it suffices to minimize

$$\sum_{\pi_j j=1,2,\ldots,J}\sum_{n\in\pi_j} D_\phi(\boldsymbol{x}_n||\boldsymbol{f}_j).$$

The last thing we need to show is $\boldsymbol{f}_j = \mu_j = \sum_{n\in\pi_j}\frac{1}{|\pi_j|}\boldsymbol{x}_n$, but this has already been proved in previous studies (Banerjee et al. 2005a, b): the best predictor in Bregman divergence is the arithmetic mean of the data. Therefore, the optimal solution $\boldsymbol{f}_j^*$ with a fixed $\mathbf{G}$ is given by

$$\boldsymbol{f}_j^* = \sum_{n:g_{nj}\neq 0}\frac{1}{|\boldsymbol{g}_j|}\boldsymbol{x}_n = \sum_{n\in\pi_j}\frac{1}{|\pi_j|}\boldsymbol{x}_n = \mu_j.$$

□

Since Bregman hard clustering is applicable to various data types with appropriate choices of $\phi(x)$ (e.g., text data with KL-divergence and speech data with IS-divergence), Bregman divergence ONMF has a wider variety of applications than does the standard ONMF.

5 Performance evaluation

5.1 Datasets

We compared the performance of those algorithms for six real-life datasets and one artificial dataset. For the artificial dataset, we followed the setting in Li et al. (2012).[6] A summary of the datasets is given in Table 3.[7]

[6] The code of their sBCD-NMF algorithm (Li et al. 2012) and the data generator are available at http://www.cc.gatech.edu/grads/l/lli86/sbcd.zip.

[7] These datasets are downloadable from http://www.cad.zju.edu.cn/home/dengcai/Data/TextData.html (Cai et al. 2009).

5.2 Compared algorithms and evaluation measures

In ONMF problems, we compared the proposed HALS-ONMF with four traditional Frobenius norm ONMF algorithms. In Bregman divergence ONMF problems, we compared sBCD-ONMF with Li's KL-divergence ONMF. In addition, in order to investigate the trade-off of orthogonality and approximation in Bregman divergence, we also compared sBCD-ONMF with sBCD-NMF, although sBCD-NMF does not have an imposed orthogonality constraint. This is because sBCD-ONMF is the first and only one ONMF algorithm workable with any Bregman divergence. We used IS- ($\phi(x) = -\log x$) and β-divergence ($\frac{1}{\beta(\beta+1)}(x^{\beta+1} - (\beta+1)x + \beta)$ with $\beta = 2$) in this comparison. We measured the orthogonality and approximation accuracy. We compared all of the algorithms shown in Table 1 except for Pompili's ONMF (Pompili et al. 2012). Pompili's ONMF algorithm (Pompili et al. 2012) was not used because their algorithm attains orthogonality but not nonnegativity. It is also reported already in their own comparison (Pompili et al. 2012). We also noted that their algorithm was slowest among all the compared algorithms. For the weighting parameter α in Li's ONMF (Li et al. 2010) and Mirzal's ONMF (Mirzal 2014), we used $\alpha = 1$, because the value worked satisfactorily on most datasets.

We employed the same evaluation setting as that in Li et al. (2012). Ten trials with different initial values are conducted and the average values of measurements were shown here. We fixed the number of iterations to 100 for all of the algorithms. We evaluated the degree of approximation and the degree of orthogonality by

$$\text{Normalized Residual Value:} \quad \frac{\| \mathbf{X} - \mathbf{F}\mathbf{G}^T \|_F^2}{\| \mathbf{X} \|_F^2} \text{ (for ONMF)} \tag{26}$$

$$\text{Relative Residual Value:} \quad \log_{10} \frac{D_\phi(\mathbf{X}||\mathbf{F}\mathbf{G}^T)}{D_\phi(\mathbf{X}||\mathbf{F}_0\mathbf{G}_0^T)} \text{ (for Bregman ONMF)} \tag{27}$$

$$\text{Orthogonality:} \quad \|\mathbf{F}^T\mathbf{F} - \mathbf{I}\|_F^2 \tag{28}$$

Here, $\mathbf{F}_0$ and $\mathbf{G}_0$ are the matrices used for initialization. In addition, we evaluated the computation time (seconds), the normalized residual value (26), and the degree of orthogonality (28) for Frobenius norm ONMF. For Bregman divergence ONMF, we evaluated the relative residual value (27) and (28) since (26) cannot be appropriately normalized for Bregman divergence.

5.3 Comparison on ONMF problems

Figure 1 shows the values of the normalized residual for $J = 30$ (number of components) for the six real-life datasets. The proposed HALS-ONMF converges faster than do the other ONMF algorithms. HALS-ONMF converges before 250 seconds for all six datasets. This is because HALS-ONMF needs a smaller number of iterations, because of the fact that HALS-ONMF solves vector-wise problems with the analytical solutions [(8) and (9)]. Figure 2 shows the degrees of orthogonality attained. The HALS-ONMF achieves almost the highest degree of orthogonality among the algorithms in the early stage, though the final degree of orthogonality is slightly worse than that of others. In the ORL dataset (Fig. 2e), a dense dataset, only HALS-ONMF succeeded in achieving an acceptable degree of orthogonality.

To show the speed of convergence, we defined the stopping criterion of the iteration according to the way conventional researches adopted (Pompili et al. 2012; Kim and Park 2008) as:

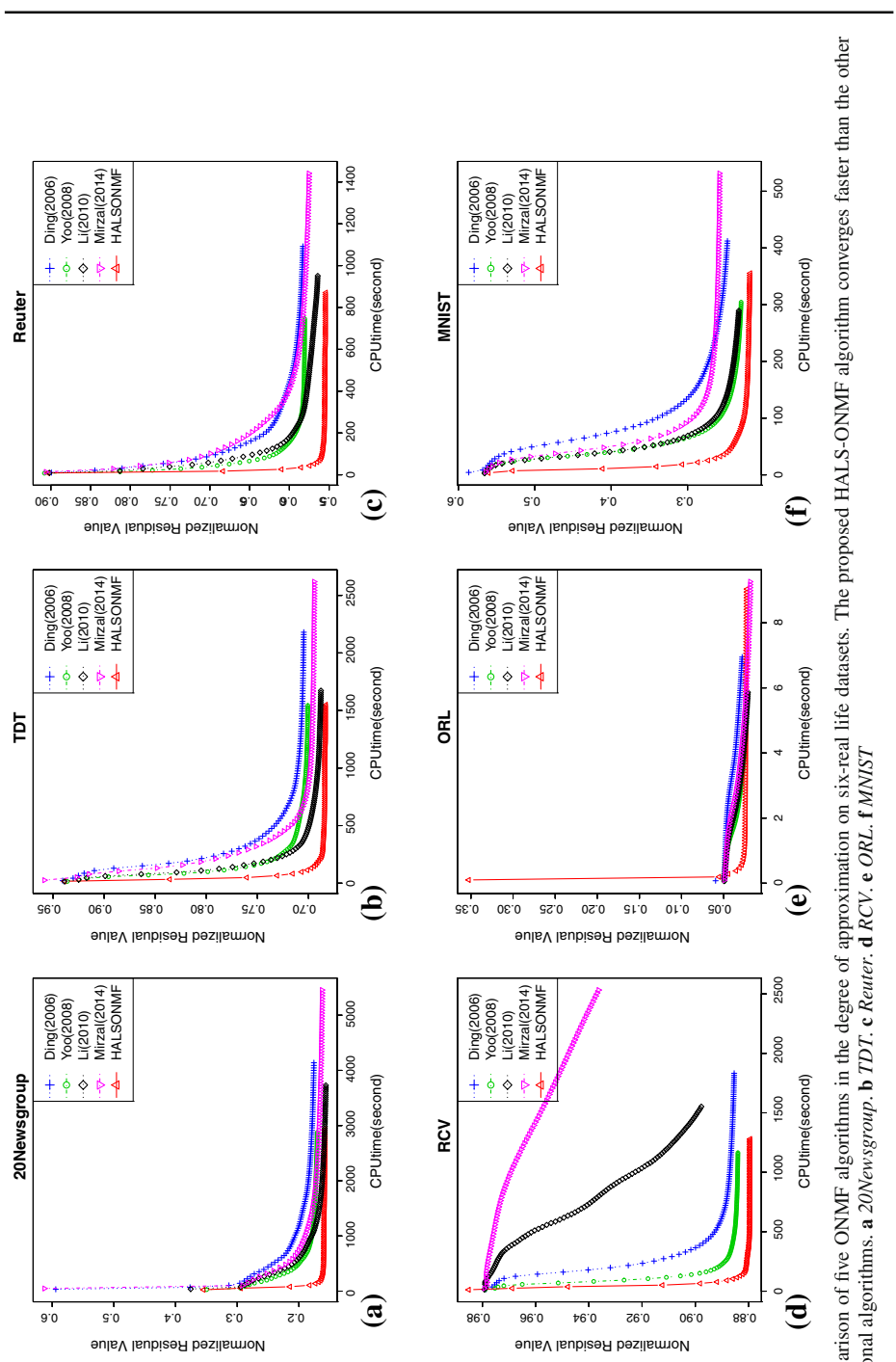

Fig. 1 Comparison of five ONMF algorithms in the degree of approximation on six-real life datasets. The proposed HALS-ONMF algorithm converges faster than the other four conventional algorithms. **a** *20Newsgroup*. **b** *TDT*. **c** *Reuter*. **d** *RCV*. **e** *ORL*. **f** *MNIST*

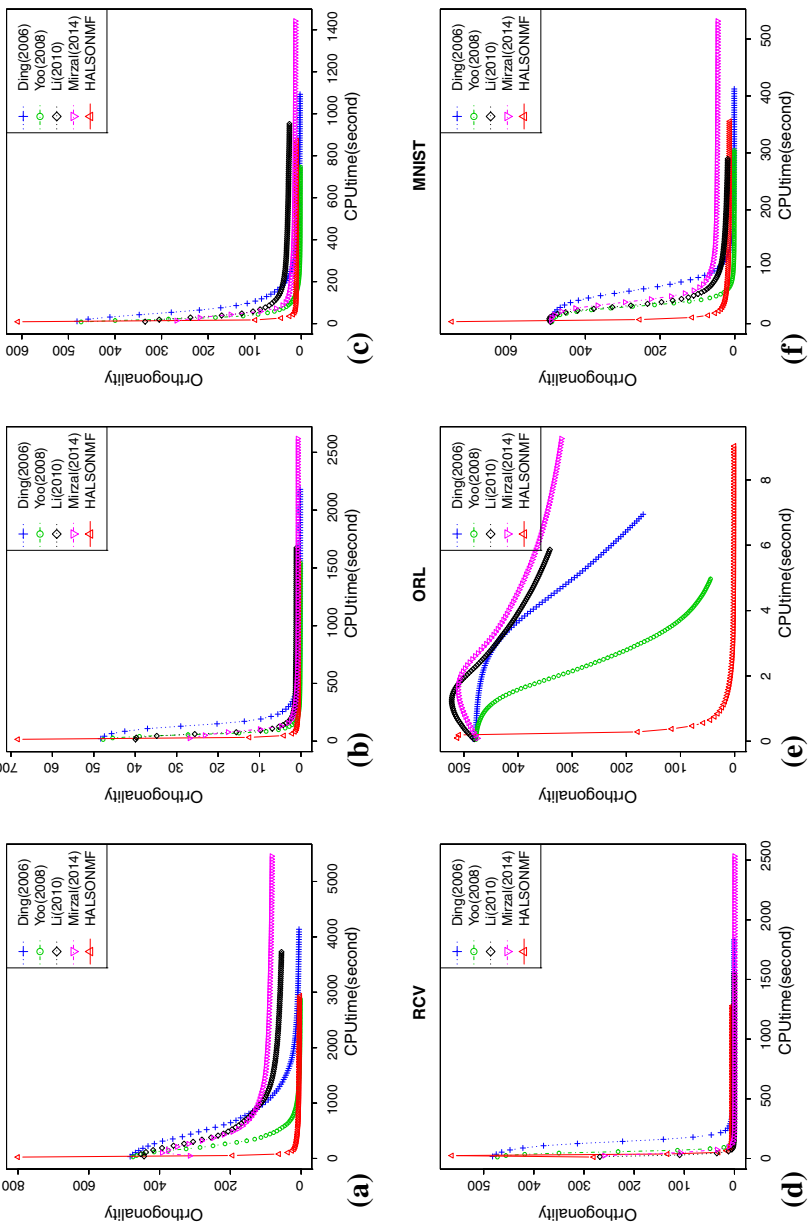

Fig. 2 Orthogonality attained by three ONMF algorithms. The proposed HALS algorithm converges faster than the other four conventional algorithms, but the final state is worse than some of the others. **a** *20Newsgroup*. **b** *TDT*. **c** *Reuter*. **d** *RCV*. **e** *ORL*. **f** *MNIST*

Table 4 The results when the stopping criterion (29) with $\epsilon = 10^{-4}$ is applied on four datasets

Algorithm	Evaluation criterion			
	# Iteration	CPU time(s)	NRV	Orthogonality
Dataset: *20Newsgroup* ($\epsilon = 10^{-4}$)				
Ding et al. (2006)	89	3638.784	0.177	6.960
Yoo and Choi (2008)	65	1899.256	0.173	**3.606**
Li et al. (2010)	83	3101.354	**0.157**	58.566
Mirzal (2014)	78	4289.880	0.164	85.314
HALSONMF	**16**	**459.936**	0.159	9.642
Dataset: *TDT* ($\epsilon = 10^{-4}$)				
Ding et al. (2006)	68	1471.580	0.706	1.957
Yoo and Choi (2008)	66	1008.414	0.702	**1.242**
Li et al. (2010)	68	1127.494	0.689	11.228
Mirzal (2014)	48	1267.000	0.697	8.031
HALSONMF	**20**	**326.622**	**0.685**	5.441
Dataset: *Reuter* ($\epsilon = 10^{-4}$)				
Ding et al. (2006)	93	1014.908	0.584	2.751
Yoo and Choi (2008)	52	383.856	0.582	**2.221**
Li et al. (2010)	100	951.008	0.564	25.394
Mirzal (2014)	93	1337.524	0.576	13.128
HALSONMF	**23**	**197.200**	**0.556**	9.825
Dataset: *MNIST* ($\epsilon = 10^{-4}$)				
Ding et al. (2006)	100	412.206	0.248	1.469
Yoo and Choi (2008)	94	285.862	0.230	**1.312**
Li et al. (2010)	100	289.862	0.233	19.011
Mirzal (2014)	68	360.948	0.261	8.031
HALSONMF	**48**	**170.702**	**0.221**	5.441

The bold values indicate the best performance among compared methods
NRV normalized residual values (26)

$$\frac{\| \mathbf{X} - \mathbf{F}^{t-1}\mathbf{G}^{t-1T} \|_F^2 - \| \mathbf{X} - \mathbf{F}^{t}\mathbf{G}^{tT} \|_F^2}{\| \mathbf{X} \|_F^2} < \epsilon, \tag{29}$$

where ϵ is a threshold and $\mathbf{G}^t$ and $\mathbf{F}^t$ are matrices after tth update. In this paper, we set the threshold ϵ to 10^{-4} in all datasets.

Table 4 shows the result on four datasets when we terminated the calculation with the stopping criterion (29).[8] The proposed HALS-ONMF is the fastest with the smallest number of iterations. The proposed algorithm converged about 1.6 to 4.1 times faster than the others, keeping comparable approximation accuracy and orthogonality.

[8] We omit the results on *RCV* dataset and *ORL* dataset because, on *RCV* dataset, Li's ONMF (Li et al. 2010) and Mirzal's ONMF (Mirzal 2014) and, on *ORL* dataset, all ONMF algorithms except the proposed HALS-ONMF decreased the approximation error too slowly and made the stopping criterion (29) satisfied in only a few iterations. See Fig. 1d, e.

5.4 Comparison on Bregman divergence ONMF problems

In Bregman divergence ONMF problems, we compared sBCD-ONMF with Li's KL-divergence ONMF (Li et al. 2010) and sBCD-NMF with KL-divergence. In addition, we compared sBCD-ONMF with sBCD-NMF for IS- or β-divergence. Unfortunately, since KL-divergence and IS-divergence do not allow zero values ($0 \notin dom_\phi$), most datasets were not suitable for this comparison. Besides, sBCD-NMF or sBCD-ONMF does not scale because of their high computational costs [see, for example, Step (11) and Step (17) in Algorithm 2]. Therefore, we dealt with only one artificial dataset of $\mathbf{X} \in \mathbb{R}^{2000\times 1000}$.

The results are shown in Fig. 3. As predicted, the sBCD-NMF algorithm without an orthogonal constraint achieved better approximation than did the algorithms with orthogonal constraints, Li's ONMF and sBCD-ONMF, while the latter two achieved a higher degree of orthogonality. In comparison of convergence speeds, sBCD-ONMF is almost the same as sBCD-NMF or even faster. Li's ONMF is inferior to sBCD-ONMF in convergence speed.[9]

In total, we can say that sBCD-ONMF is a fast algorithms to find a solution in Bregman divergence ONMF problems with a sufficient degree of orthogonality at the expense of a little amount of degradation of approximation.

5.5 Clustering experiments

As we stated before, ONMF is suitable for clustering tasks more than standard NMF. This is because the constrained matrix $\mathbf{F}$ can be considered as an indicator matrix in ONMF. Let $\mathbf{X}$ be an $instance \times feature$ matrix factorized by $\mathbf{FG}^T$. Then ith row of $\mathbf{F}$ can be considered as a membership vector of instance i to J groups (features). Especially, a solution $\mathbf{FG}^T$ in ONMF is expected to have a crisp membership of a single one. We assign the ith instance to kth cluster such as

$$k = \underset{j}{\operatorname{argmax}}\, \mathbf{F}_{ij}.$$

We compared the proposed HALS-ONMF and sBCD-ONMF with one standard NMF algorithm (HALS-NMF) and conventional ONMF algorithms [Ding's ONMF (2006) and Yoo's ONMF (2008)]. We set the number iteration to 30 which was sufficient for convergence in the previous experiments in Sects. 5.3 and 5.4. In addition, we conducted k-means algorithm as a base-line method. We used four TREC document classification datasets (see Table 5 for the detail). Since these dataset have class labels, we hid them for clustering and then evaluated the difference between the true clustering induced by the class labels and the obtained clustering.

We measured Normalized Mutual Information (NMI) defined as

$$\text{NMI} : \frac{I(\hat{C}; C)}{(H(\hat{C}) + H(C))/2},$$

where $\hat{C}$ is the predicted clustering and C is the ground truth. Here, $H(\cdot)$ is Shanon Entropy, and $I(;)$ is the Mutual Information. We averaged the results for ten trials with different initial points.

[9] In Li et al. (2010), it is reported that Li's KL-divergence ONMF algorithm needs a large number of iterations to attain a sufficient level of orthogonality.

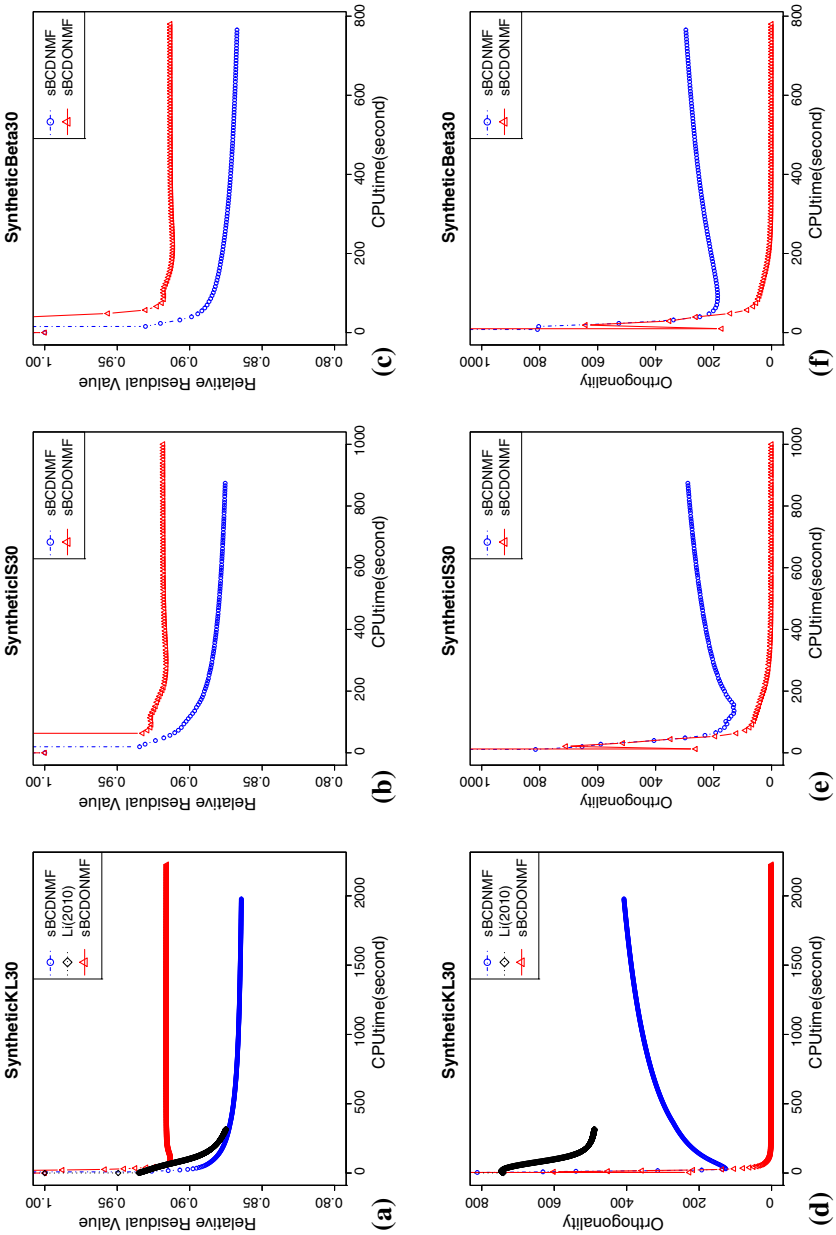

Fig. 3 Comparison of KL-divergence, IS-divergence and β-divergence NMF on an artificial dataset ($\mathbf{X} \in \mathbb{R}^{2000\times1000}$). The first row shows the value of relative residual and the second row shows the degree of orthogonality. **a** KL-divergence. **b** IS-divergence. **c** β-Divergence. **d** KL-divergence. **e** IS-divergence. **f** β-Divergence

Table 5 Datasets used in the clustering experiments

Dataset	# Instance	# Feature	# Class
k1b	2340	21,819	6
Reviews	4069	18,483	5
Sports	8580	14,870	7
Hitech	2301	10,080	7

Table 6 The clustering results evaluated in NMI (the higher, the better) on four-real life datasets

Criterion	Algorithm	Dataset			
		k1b	Reviews	Sports	Hitech
	K-means	0.503 ± 0.018	0.338 ± 0.040	0.235 ± 0.022	0.197 ± 0.019
Frobenius norm	HALS-NMF (Cichocki et al. 2009; Li et al. 2012)	0.527 ± 0.081	0.337 ± 0.040	0.268 ± 0.024	0.299 ± 0.020
	Ding's ONMF (Ding et al. 2006)	0.520 ± 0.039	0.333 ± 0.082	0.242 ± 0.029	0.292 ± 0.017
	Yoo's ONMF (Yoo and Choi 2008)	0.559 ± 0.035	0.370 ± 0.065	0.271 ± 0.040	0.315 ± 0.018
	HALS-ONMF	0.540 ± 0.050	0.420 ± 0.095	0.390 ± 0.048	$\mathbf{0.334 \pm 0.015}$
KL-divergence	sBCD-NMF (Li et al. 2012)	$\mathbf{0.593 \pm 0.032}$	$\mathbf{0.530 \pm 0.071}$	0.544 ± 0.032	0.271 ± 0.048
	SBCD-ONMF	0.587 ± 0.019	0.474 ± 0.095	$\mathbf{0.610 \pm 0.040}$	0.228 ± 0.017
IS-divergence	sBCD-NMF (Li et al. 2012)	0.032 ± 0.045	0.022 ± 0.017	0.070 ± 0.059	0.016 ± 0.017
	SBCD-ONMF	0.025 ± 0.010	0.048 ± 0.024	0.030 ± 0.020	0.034 ± 0.020

The bold values indicate the best performance among compared methods

Unfortunately, the general advantage of ONMF over NMF was not confirmed,[10] as long as their algorithms are with Frobenius norm. Nevertheless, the proposed HALS-ONMF achieved the best score in NMI among them. Rather, we confirmed the advantage of KL-divergence over Frobenius norm and IS-divergence. This is not an unexpected result because it is known that the document data is well explained by Multinomial distribution models and minimizing KL-divergence is corresponding to maximum likelihood with Multinomial distribution model (Banerjee et al. 2005b; Li et al. 2012). The best choice is one of conventional SBCD-NMF with KL-divergence, sBCD-ONMF with KL-divergence, and HALS-ONMF with Frobenius norm (Table 6).

6 Conclusion

In this paper, we have proposed a fast algorithm for solving one-sided orthogonal nonnegative matrix factorization problems in the Frobenius norm and in Bregman divergence. Orthogonal NMF algorithms proposed so far suffered from slow convergence mainly due to their matrix-wise updates. By decomposing the matrix-type orthogonality condition into a set of column-

[10] Note that the orthogonal constraint gives more crisp membership, however, this does not mean better clustering accuracy.

wise orthogonality conditions, we succeeded in speeding up the convergence. One of the proposed algorithms is the first algorithm to solve a Bregman divergence NMF problem with an orthogonal constraint. In addition, we showed that Bregman divergence ONMF problem is equivalent to Bregman hard clustering. Experiments for six real-life datasets and an artificial dataset demonstrated that the proposed algorithms are in fact faster than state-of-the-art algorithms in convergence while keeping a satisfactory level of orthogonality. In the best case, the proposed algorithm converged more than four times faster than state-of-the-art algorithms.

Acknowledgments This work was partially supported by JSPS KAKENHI Grant Numbers 14J01495 and 15H02719.

References

Banerjee, A., Guo, X., & Wang, H. (2005). On the optimality of conditional expectation as a Bregman predictor. *IEEE Transactions on Information Theory*, *51*(7), 2664–2669.

Banerjee, A., Merugu, S., Dhillon, I. S., & Ghosh, J. (2005). Clustering with Bregman divergences. *The Journal of Machine Learning Research*, *6*, 1705–1749.

Cai, D., He, X., Han, J., & Huang, T. S. (2011). Graph regularized nonnegative matrix factorization for data representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *33*(8), 1548–1560.

Cai, D., Wang, X., & He, X. (2009). Probabilistic dyadic data analysis with local and global consistency. In *Proceedings of the 26th annual international conference on machine learning* (pp. 105–112). ACM.

Cichocki, A., & Anh-Huy, P. (2009). Fast local algorithms for large scale nonnegative matrix and tensor factorizations. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, *92*(3), 708–721.

Cichocki, A., Zdunek, R., Phan, A. H., & Amari, S.-I. (2009). *Nonnegative matrix and tensor factorizations: Applications to exploratory multi-way data analysis and blind source separation*. New York: Wiley.

Ding, C., Li, T., & Peng, W. (2008). On the equivalence between non-negative matrix factorization and probabilistic latent semantic indexing. *Computational Statistics & Data Analysis*, *52*(8), 3913–3927.

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

Hsieh, C.-J., & Dhillon, I. S. (2011). Fast coordinate descent methods with variable selection for non-negative matrix factorization. In *Proceedings of the 17th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 1064–1072). ACM.

Kim, J., He, Y., & Park, H. (2014). Algorithms for nonnegative matrix and tensor factorizations: A unified view based on block coordinate descent framework. *Journal of Global Optimization*, *58*(2), 285–319.

Kim, J., & Park, H. (2008). Toward faster nonnegative matrix factorization: A new algorithm and comparisons. In *Proceedings of the 8th IEEE international conference on data mining* (pp. 353–362). IEEE.

Kim, J., & Park, H. (2011). Fast nonnegative matrix factorization: An active-set-like method and comparisons. *SIAM Journal on Scientific Computing*, *33*(6), 3261–3281.

Lee, D. D., & Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, *13*, 556–562.

Li, L., Lebanon, G., & Park, H. (2012). Fast Bregman divergence NMF using Taylor expansion and coordinate descent. In *Proceedings of the 18th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 307–315). ACM.

Li, Z., Wu, X., & Peng, H. (2010). Nonnegative matrix factorization on orthogonal subspace. *Pattern Recognition Letters*, *31*(9), 905–911.

Lin, C.-J. (2007). Projected gradient methods for nonnegative matrix factorization. *Neural Computation*, *19*(10), 2756–2779.

Mirzal, A. (2014). A convergent algorithm for orthogonal nonnegative matrix factorization. *Journal of Computational and Applied Mathematics*, *260*, 149–166.

Pompili, F., Gillis, N., Absil, P.-A., & Glineur, F. (2012). Two algorithms for orthogonal nonnegative matrix factorization with application to clustering. arXiv preprint arXiv:1201.0901.

Yoo, J., & Choi, S. (2008). Orthogonal nonnegative matrix factorization: Multiplicative updates on stiefel manifolds. In *Intelligent data engineering and automated learning* (pp. 140–147). Springer.