IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 12, 2016, accepted August 12, 2016, date of publication September 2, 2016, date of current version September 28, 2016.

Digital Object Identifier 10.1109/ACCESS.2016.2605704

A Fast Non-Smooth Nonnegative Matrix Factorization for Learning Sparse Representation

ZUYUAN YANG[1], (Member, IEEE), YU ZHANG[1], WEI YAN[2], YONG XIANG[3], (Senior Member, IEEE), AND SHENGLI XIE[1], (Senior Member, IEEE)

[1]Guangdong Key Laboratory of IoT Information Technology, School of Automation, Guangdong University of Technology, Guangzhou 510006, China
[2]Faculty of Science and Technology, University of Macau, Macau 999078, China
[3]School of Information Technology, Deakin University, Melbourne, VIC 3125, Australia

Corresponding author: S. Xie (shlxie@gdut.edu.cn)

This work was supported in part by the Pearl River S&T Nova Program of Guangzhou under Grant 201610010196, in part by the Guangdong Natural Science Funds for Distinguished Young Scholar under Grant 2014A030306037, in part by the National Natural Science Foundation of China under Grant U1501251 and Grant 61333013, in part by the Program for New Century Excellent Talents in University under Grant NCET-13-0740, in part by the Guangdong Program for support of Top-Notch Young Professionals under Grant 2014TQ01X144, and in part by the Guangdong Funds for Excellent Doctoral Dissertations Scholar under Grant sybzzxm201214.

ABSTRACT Nonnegative matrix factorization (NMF) is a hot topic in machine learning and data processing. Recently, a constrained version, non-smooth NMF (NsNMF), shows a great potential in learning meaningful sparse representation of the observed data. However, it suffers from a slow linear convergence rate, discouraging its applications to large-scale data representation. In this paper, a fast NsNMF (FNsNMF) algorithm is proposed to speed up NsNMF. In the proposed method, it first shows that the cost function of the derived sub-problem is convex and the corresponding gradient is Lipschitz continuous. Then, the optimization to this function is replaced by solving a proximal function, which is designed based on the Lipschitz constant and can be solved through utilizing a constructed fast convergent sequence. Due to the usage of the proximal function and its efficient optimization, our method can achieve a nonlinear convergence rate, much faster than NsNMF. Simulations in both computer generated data and the real-world data show the advantages of our algorithm over the compared methods.

INDEX TERMS Nonnegative matrix factorization, sparse representation, nonlinear convergence rate.

I. INTRODUCTION

Nonnegative matrix factorization (NMF) is a classic problem and it is recently popularized by Lee and Seung [1]. It aims to decompose a nonnegative matrix $\mathbf{V}$ into the product of two low-rank nonnegative factor matrices $\mathbf{W}$ and $\mathbf{H}$ approximately, i.e., $\mathbf{V} \approx \mathbf{WH}$. This provides a powerful tool to learn dimensionality reduced and part-based data representation. As a result, it has attracted seamless attention during these years [2]–[7].

Regarding the NMF algorithms, a simple method using multiplicative update rule is introduced in [8]. It is easy to implement but converges slowly, as the used gradient is of first order. Yet, its solution is not necessarily a stationary point. To improve this method, the active set based algorithm is provided to speed up convergence in [9], but it faces the numerical instability problem. On the other hand, the projected gradient based method in [10] can get a stable solution. However, this method still suffers from a slow convergence rate. There exist some other advanced NMF methods, such as the accelerated schemes [11], the symmetric NMF [12], [13], the separable NMF [14], and the low-rank factorization algorithms [15], [16]. A common problem of the methods in [8]–[16] is that they does not necessarily generate a needed solution.

To obtain the factor matrices with desired features, the constrained NMF algorithms are developed for different applications. In [17], NMF with sparse constraint is given to generate solution with different sparsity levels, which is quite useful in data analysis. In [18], the volume constraint based NMF is designed for solving blind source separation problem. In [19], the label information constraint based NMF is applied to image representation, and the regularized NMF algorithm is summarized in [20].

Recently, a new constrained version, non-smooth NMF (NsNMF) [21], shows a great potential in learning meaningful representations of the collected data. It generates

sparse solutions by using a non-smooth factor. Also, it can decompose the given data into nonoverlapping parts, more meaningful than traditional sparsity. However, the convergence speed of this method is slow, as it utilizes the traditional multiplicative update rule in [8].

In this paper, a fast NsNMF (FNsNMF) method is proposed to overcome the aforementioned slow convergence problem. In the proposed method, the widely used alternative optimization scheme is employed, i.e., optimizing one matrix factor with another fixed [6]. For the optimization of each factor, we show that the cost function is convex and its gradient is Lipschitz continuous. Then, a proximal function of the object is designed based on the Lipschitz constant. After that, a fast convergent sequence is constructed to solve this function, and finally obtain the optimal solution to the original cost function. Based on the analysis in [22]–[24], the used scheme can achieve the nonlinear convergence rate $O(\frac{1}{k^2})$, greatly speeding up NsNMF with linear convergence rate.

The remainder of this paper is organized as follows. The related NMF models are introduced in Section II, together with the details of the proposed FNsNMF algorithm. In section III, simulation results are provided to illustrate the performance of the proposed algorithm and the compared methods. And conclusions are given in Section IV.

II. FAST NON-SMOOTH NONNEGATIVE MATRIX FACTORIZATION

A new method with nonlinear convergence rate is proposed for solving NsNMF in this Section. We start from the introduction of the NMF and NsNMF models.

A. MODELS

For a given nonnegative matrix $\mathbf{V} \in \mathbf{R}_+^{m\times n}$, NMF aims to find two low-rank nonnegative factor matrices $\mathbf{W} \in \mathbf{R}_+^{m\times r}$ and $\mathbf{H} \in \mathbf{R}_+^{r\times n}$ with $r < \min\{m, n\}$, such that their product approximates to $\mathbf{V}$. Mathematically, one often gets $\mathbf{W}$ and $\mathbf{H}$ by solving the following Euclidean distance based minimization problem [1], [17]:

$$\min : \quad \frac{1}{2}\|\mathbf{V} - \mathbf{WH}\|_F^2 \tag{1}$$

where $\|\cdot\|_F$ denotes the matrix Frobenius norm.

To get sparse solutions for more meaningful data representation, a novel scheme is utilized in [21]. In this method, a nonnegative and symmetric non-smoothness factor matrix $\mathbf{S} \in \mathbf{R}_+^{r\times r}$ is embedded between $\mathbf{W}$ and $\mathbf{H}$. Based on this factor, an attractive model is constructed as

$$\min : \quad f(\mathbf{W}, \mathbf{H}) = \frac{1}{2}\|\mathbf{V} - \mathbf{WSH}\|_F^2 \tag{2}$$

with

$$\mathbf{S} = (1-\theta)\,\mathbf{I} + \frac{\theta}{r}\mathbf{1}\mathbf{1}^T \tag{3}$$

where $\mathbf{I}$ is the identity matrix, $\mathbf{1}$ is the vector of ones, $\theta \in [0, 1]$ is a parameter [21], and the superscript T denotes transpose. Clearly, (2) is equivalent to (1) in the case $\theta = 0$. It is interesting that the three-factors style of (2) is similar to the model in [25]. However, the constraints to the middle factor are different in these two models.

For the optimization of (2), one often utilizes the alternative least square scheme, i.e., optimizing one factor while fixing the other. For notational simplicity, we write $f(\mathbf{W}, \mathbf{H})$ as $f_1(\mathbf{H})$ when $\mathbf{W}$ is fixed, and as $f_2(\mathbf{W})$ when $\mathbf{H}$ is fixed. As for the optimization of $f_1(\mathbf{H})$ (or $f_2(\mathbf{W})$), it is natural to use traditional active set method or multiplicative update rule [8], [9]. However, they suffer from either numerical instability or a slow convergence rate [7]. To design more efficient algorithm for minimizing $f_1(\mathbf{H})$ and $f_2(\mathbf{W})$, we further analyze the features of these functions. It is known that they are convex for a given $\mathbf{S}$ [8]. Moreover, we have the following lemma.

Lemma 1: The gradient of $f_1(\mathbf{H})$ is Lipschitz continuous and the Lipschitz constant is $L = \left\|\mathbf{S}^T\mathbf{W}^T\mathbf{WS}\right\|_2$.

Proof: According to (2), we can calculate the gradient of $f_1(\mathbf{H})$ (i.e., $f(\mathbf{W}, \mathbf{H})$ with fixing $\mathbf{W}$) by

$$\begin{aligned}\nabla_{\mathbf{H}} f_1(\mathbf{H}) &= (\mathbf{WS})^T\mathbf{WSH} - (\mathbf{WS})^T\mathbf{V} \\ &= \mathbf{S}^T\mathbf{W}^T\mathbf{WSH} - \mathbf{S}^T\mathbf{W}^T\mathbf{V}.\end{aligned} \tag{4}$$

Then, for any two matrices $\mathbf{H}_1, \mathbf{H}_2 \in \mathbf{R}_+^{r\times n}$, one can derive that

$$\begin{aligned}&\|\nabla_{\mathbf{H}} f_1(\mathbf{H}_1) - \nabla_{\mathbf{H}} f_1(\mathbf{H}_2)\|_F^2 \\ &= \left\|\mathbf{S}^T\mathbf{W}^T\mathbf{WS}(\mathbf{H}_1 - \mathbf{H}_2)\right\|_F^2 \\ &= tr\left((\mathbf{U}\Sigma\mathbf{U}^T(\mathbf{H}_1 - \mathbf{H}_2))^T(\mathbf{U}\Sigma\mathbf{U}^T(\mathbf{H}_1 - \mathbf{H}_2))\right)\end{aligned} \tag{5}$$

where $\mathbf{U}\Sigma\mathbf{U}^T$ is the singular value decomposition of $\mathbf{S}^T\mathbf{W}^T\mathbf{WS}$, and tr denotes trace. Let the largest singular value be δ. By some algebra, it can be derived from (5) that

$$\begin{aligned}&\|\nabla_{\mathbf{H}} f_1(\mathbf{H}_1) - \nabla_{\mathbf{H}} f_1(\mathbf{H}_2)\|_F^2 \\ &= tr\left(\mathbf{U}^T(\mathbf{H}_1 - \mathbf{H}_2)(\mathbf{H}_1 - \mathbf{H}_2)^T\mathbf{U}\Sigma^2\right) \\ &\le \delta^2 tr\left(\mathbf{U}^T(\mathbf{H}_1 - \mathbf{H}_2)(\mathbf{H}_1 - \mathbf{H}_2)^T\mathbf{U}\right) \\ &= \delta^2\|\mathbf{H}_1 - \mathbf{H}_2\|_F^2\end{aligned} \tag{6}$$

where the last two equations come from the fact that $\mathbf{U}^T\mathbf{U}$ and $\mathbf{U}\mathbf{U}^T$ are identity matrices. From (6), we can find a constant L (e.g., L=δ), such that

$$\|\nabla_{\mathbf{H}} f_1(\mathbf{H}_1) - \nabla_{\mathbf{H}} f_1(\mathbf{H}_2)\|_F \le L\|\mathbf{H}_1 - \mathbf{H}_2\|_F .$$

Therefore, $\nabla_{\mathbf{H}} f_1(\mathbf{H})$ is Lipschitz continuous and the Lipschitz constant is the largest singular value of $\mathbf{S}^T\mathbf{W}^T\mathbf{WS}$, i.e., $L = \left\|\mathbf{S}^T\mathbf{W}^T\mathbf{WS}\right\|_2$. This completes the proof. ■

Regarding $f_2(\mathbf{W})$, according to (2), it can be written by $\frac{1}{2}\left\|\mathbf{V}^T - \mathbf{H}^T\mathbf{S}^T\mathbf{W}^T\right\|_F^2$. Then, similar to Lemma 1, we can conclude that the gradient of $f_2(\mathbf{W})$ is Lipschitz continuous with Lipschitz constant $N = \left\|\mathbf{SHH}^T\mathbf{S}^T\right\|_2$ (leaving out the proof for saving space).

B. FNsNMF ALGORITHM

Since the optimizations to $f_1(\mathbf{H})$ and $f_2(\mathbf{W})$ are symmetric, we mainly analyze how to optimize $f_1(\mathbf{H})$ here. Recall that $f_1(\mathbf{H})$ is convex and its gradient is Lipschitz continuous. Based on these conclusions and the analysis in [22]–[24], a fast algorithm with nonlinear convergence rate is designed below.

First, we construct the following proximal function $\phi_1(\mathbf{Y}, \mathbf{H})$ of $f_1(\mathbf{H})$ on $\mathbf{Y}$:

$$\phi_1(\mathbf{Y}, \mathbf{H}) = f_1(\mathbf{Y}) + \langle \nabla_{\mathbf{H}} f_1(\mathbf{Y}), \mathbf{H} - \mathbf{Y} \rangle + \frac{L}{2} \|\mathbf{H} - \mathbf{Y}\|_F^2 \tag{7}$$

where $L = \left\| \mathbf{S}^T \mathbf{W}^T \mathbf{W} \mathbf{S} \right\|_2$ is the Lipschitz constant and $\langle \cdot, \cdot \rangle$ denotes the matrix inner product. To get the optimal $\mathbf{H}$, denoted by $\hat{\mathbf{H}}$, for the minimization of $\phi_1(\mathbf{Y}, \mathbf{H})$, the Lagrange multiplier method is applied. Regarding $\hat{\mathbf{H}}$, the Karush-Kuhn-Tucker (K.K.T.) conditions of (7) are

$$\begin{cases} \nabla_{\mathbf{H}} \phi_1(\mathbf{Y}, \hat{\mathbf{H}}) \geq 0 \\ \hat{\mathbf{H}} \geq 0 \\ \nabla_{\mathbf{H}} \phi_1(\mathbf{Y}, \hat{\mathbf{H}}) \otimes \hat{\mathbf{H}} = 0 \end{cases} \tag{8}$$

where $\otimes$ denotes the component-wise multiplication. Since $\nabla_{\mathbf{H}} \phi_1(\mathbf{Y}, \hat{\mathbf{H}}) = \nabla_{\mathbf{H}} f_1(\mathbf{Y}) + L(\hat{\mathbf{H}} - \mathbf{Y})$, then motivated by [10], $\hat{\mathbf{H}}$ can be calculated by

$$\hat{\mathbf{H}} = \mathbf{P}(\mathbf{Y} - \frac{1}{L} \nabla_{\mathbf{H}} f_1(\mathbf{Y})) \tag{9}$$

where $\mathbf{P}(\mathbf{Z})$ projects all negative entries of $\mathbf{Z}$ to zero.

We further show that $\hat{\mathbf{H}}$ in (9) satisfies the needed K.K.T. condition given in (8). The analysis includes two cases below:
1) $\mathbf{Y} - \frac{1}{L} \nabla_{\mathbf{H}} f_1(\mathbf{Y}) \geq 0$: in this case, (9) degenerates into

$$\begin{aligned} \hat{\mathbf{H}} &= \mathbf{Y} - \frac{1}{L} \nabla_{\mathbf{H}} f_1(\mathbf{Y}) \\ &\geq 0. \end{aligned} \tag{10}$$

Substituting (10) to $\nabla_{\mathbf{H}} \phi_1(\mathbf{Y}, \hat{\mathbf{H}})$, it holds that

$$\begin{aligned} &\nabla_{\mathbf{H}} \phi_1(\mathbf{Y}, \hat{\mathbf{H}}) \\ &= \nabla_{\mathbf{H}} f_1(\mathbf{Y}) + L(\hat{\mathbf{H}} - \mathbf{Y}) \\ &= \nabla_{\mathbf{H}} f_1(\mathbf{Y}) + L(\mathbf{Y} - \frac{1}{L} \nabla_{\mathbf{H}} f_1(\mathbf{Y}) - \mathbf{Y}) \\ &= 0. \end{aligned} \tag{11}$$

From (10) and (11), one can conclude that (8) is satisfied.
2) $\mathbf{Y} - \frac{1}{L} \nabla_{\mathbf{H}} f_1(\mathbf{Y}) < 0$: in this case, (9) is equivalent to $\hat{\mathbf{H}} = 0$. Substituting it to $\nabla_{\mathbf{H}} \phi_1(\mathbf{Y}, \hat{\mathbf{H}})$, it yields

$$\begin{aligned} &\nabla_{\mathbf{H}} \phi_1(\mathbf{Y}, \hat{\mathbf{H}}) \\ &= \nabla_{\mathbf{H}} f_1(\mathbf{Y}) + L(\hat{\mathbf{H}} - \mathbf{Y}) \\ &= \nabla_{\mathbf{H}} f_1(\mathbf{Y}) - L\mathbf{Y} \\ &> 0. \end{aligned} \tag{12}$$

One can see that (8) is satisfied again. Thus, the result calculated by (9) satisfies the needed K.K.T. condition. Alternatively, (9) corresponds to the optimal solution for minimizing the cost function $\phi_1(\mathbf{Y}, \mathbf{H})$ with respect to $\mathbf{H}$.

Based on the constructed $\phi_1(\mathbf{Y}, \mathbf{H})$ above, we introduce how to get $\mathbf{H}$ for minimizing $f_1(\mathbf{H})$. From (9) and the schemes in [23], at the kth iteration, we have

$$\begin{cases} \mathbf{H}_k = \mathbf{P}(\mathbf{Y}_{k-1} - \frac{1}{L} \nabla_{\mathbf{H}} f_1(\mathbf{Y}_{k-1})) \\ \beta_k = \frac{1 + \sqrt{4\beta_{k-1}^2 + 1}}{2} \\ \mathbf{Y}_k = \mathbf{H}_k + \frac{\beta_{k-1} - 1}{\beta_k} (\mathbf{H}_k - \mathbf{H}_{k-1}). \end{cases} \tag{13}$$

Given initial values $\mathbf{H}_0$, $\beta_0 = 1$, and let $\mathbf{Y}_0 = \mathbf{H}_0$, one can alternatively update $\mathbf{H}_k$, β_k, and $\mathbf{Y}_k$ using (13). As for the stop criterion, the gradient based tolerance in [10] can be utilized directly. Denoting the final $\mathbf{H}_k$ to be $\mathbf{H}^*$, the optimal $\mathbf{H}$ is obtained by

$$\mathbf{H} = \mathbf{H}^*. \tag{14}$$

Regarding the optimization of $f_2(\mathbf{W})$, similar to the approach above, one can first construct the following proximal function $\phi_2(\mathbf{X}, \mathbf{W})$ of $f_2(\mathbf{W})$ on $\mathbf{X}$:

$$\phi_2(\mathbf{X}, \mathbf{W}) = f_2(\mathbf{X}) + \langle \nabla_{\mathbf{W}} f_2(\mathbf{X}), \mathbf{W} - \mathbf{X} \rangle + \frac{N}{2} \|\mathbf{W} - \mathbf{X}\|_F^2 \tag{15}$$

and then get the corresponding $\mathbf{W}^*$ through updating the sequences below:

$$\begin{cases} \mathbf{W}_k = \mathbf{P}(\mathbf{X}_{k-1} - \frac{1}{N} \nabla_{\mathbf{W}} f_2(\mathbf{X}_{k-1})) \\ \alpha_k = \frac{1 + \sqrt{4\alpha_{k-1}^2 + 1}}{2} \\ \mathbf{X}_k = \mathbf{W}_k + \frac{\alpha_{k-1} - 1}{\alpha_k} (\mathbf{W}_k - \mathbf{W}_{k-1}). \end{cases} \tag{16}$$

Finally, the optimal $\mathbf{W}$ is obtained by

$$\mathbf{W} = \mathbf{W}^*. \tag{17}$$

Based on the analysis above, the proposed FNsNMF algorithm is summarized as follows:

Step 1: Initialization: give a desired θ and a stop criterion (e.g., the gradient based tolerance [10]).

Step 2: Update $\mathbf{H}$ *by* (14).

Step 3: Update $\mathbf{W}$ *by* (17).

Step 4: If the stop criterion is not satisfied, goto Step 2; otherwise, end.

C. CONVERGENCE RATE AND COMPUTATIONAL COMPLEXITY

In the proposed algorithm, it utilizes the traditional alternative iteration scheme for the update of $\mathbf{H}$ and $\mathbf{W}$. We first focus on the convergence in calculating the optimal $\mathbf{H}$ and give the following Proposition:

Proposition 1: Given initial matrix $\mathbf{H}_0$ and sequence $\{\mathbf{H}_k\}_{k=2}^{\infty}$ generated by (13), it holds that

$$f_1(\mathbf{H}_k) - f_1(\mathbf{H}^*) \leq \frac{2L||\mathbf{H}_0 - \mathbf{H}^*||_F^2}{(k+1)^2}. \tag{18}$$

where $\mathbf{H}^*$ denotes the optimal solution of (2) in the case that $\mathbf{W}$ is fixed and $L = \left\|\mathbf{S}^T\mathbf{W}^T\mathbf{W}\mathbf{S}\right\|_2$ is the Lipschitz constant.

Proof: Based on [22, Th. 2.2.7], given $\hat{\mathbf{H}}$ which minimizes $\phi_1(\mathbf{Y}, \mathbf{H})$, for any $\mathbf{H} \in \mathbf{R}_+^{r\times n}$ and $\mathbf{Y} \in \mathbf{R}^{r\times n}$, it holds that

$$f_1(\mathbf{H}) \geq f_1(\hat{\mathbf{H}}) + L\langle\hat{\mathbf{H}} - \mathbf{Y}, \mathbf{Y} - \mathbf{H}\rangle + \frac{L}{2}||\mathbf{Y} - \hat{\mathbf{H}}||_F^2. \quad (19)$$

Let $\mathbf{Y} = \mathbf{Y}_k$, then based on the first equation in (13), we have $\hat{\mathbf{H}} = \mathbf{H}_{k+1}$. Let $\mathbf{H} = \mathbf{H}_k$, then substitute $\mathbf{Y}$, $\hat{\mathbf{H}}$, $\mathbf{H}$ above to (19), we get

$$\begin{aligned} f_1(\mathbf{H}_k) \geq\ & f_1(\mathbf{H}_{k+1}) + \frac{L}{2}||\mathbf{Y}_k - \mathbf{H}_{k+1}||_F^2 \\ & + L\langle\mathbf{H}_{k+1} - \mathbf{Y}_k, \mathbf{Y}_k - \mathbf{H}_k\rangle. \end{aligned} \quad (20)$$

Similarly, substitute $\mathbf{Y} = \mathbf{Y}_k, \hat{\mathbf{H}} = \mathbf{H}_{k+1}, \mathbf{H} = \mathbf{H}^*$ to (19), we have

$$\begin{aligned} f_1(\mathbf{H}^*) \geq\ & f_1(\mathbf{H}_{k+1}) + \frac{L}{2}||\mathbf{Y}_k - \mathbf{H}_{k+1}||_F^2 \\ & + L\langle\mathbf{H}_{k+1} - \mathbf{Y}_k, \mathbf{Y}_k - \mathbf{H}^*\rangle. \end{aligned} \quad (21)$$

Since $\beta_k > 1, \forall k > 0$, we multiply $\beta_k - 1$ to both side of (20) and add it to (21), then it holds that

$$\begin{aligned} & (\beta_k - 1)f_1(\mathbf{H}_k) + f_1(\mathbf{H}^*) \\ & \geq L\langle\mathbf{H}_{k+1} - \mathbf{Y}_k, \beta_k\mathbf{Y}_k - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*\rangle \\ & \quad + \beta_k f_1(\mathbf{H}_{k+1}) + \frac{L\beta_k}{2}||\mathbf{Y}_k - \mathbf{H}_{k+1}||_F^2. \end{aligned} \quad (22)$$

Multiplying both sides of (22) by β_k, we have

$$\begin{aligned} & (\beta_k^2 - \beta_k)f_1(\mathbf{H}_k) + \beta_k f_1(\mathbf{H}^*) \\ & \geq L\langle\beta_k(\mathbf{H}_{k+1} - \mathbf{Y}_k), \beta_k\mathbf{Y}_k - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*\rangle \\ & \quad + \beta_k^2 f_1(\mathbf{H}_{k+1}) + \frac{L\beta_k^2}{2}||\mathbf{Y}_k - \mathbf{H}_{k+1}||_F^2. \end{aligned} \quad (23)$$

From the second equation in (13), it holds $\beta_{k-1}^2 = \beta_k^2 - \beta_k$. Then, (23) can be rewritten as

$$\begin{aligned} & \beta_{k-1}^2 f_1(\mathbf{H}_k) + (\beta_k^2 - \beta_{k-1}^2)f_1(\mathbf{H}^*) - \beta_k^2 f_1(\mathbf{H}_{k+1}) \\ & \geq L\langle\beta_k(\mathbf{H}_{k+1} - \mathbf{Y}_k), \beta_k\mathbf{Y}_k - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*\rangle \\ & \quad + \frac{L}{2}||\beta_k\mathbf{H}_{k+1} - \beta_k\mathbf{Y}_k||_F^2. \end{aligned} \quad (24)$$

Notably that for any matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$, it holds $||\mathbf{B} - \mathbf{A}||_F^2 + 2\langle\mathbf{B} - \mathbf{A}, \mathbf{A} - \mathbf{C}\rangle = ||\mathbf{B} - \mathbf{C}||_F^2 - ||\mathbf{A} - \mathbf{C}||_F^2$. And from the third equation in (13), we have $\beta_k(\mathbf{Y}_k - \mathbf{H}_k) = (\beta_{k-1} - 1)(\mathbf{H}_k - \mathbf{H}_{k-1})$. Then, (24) can be rewritten as

$$\begin{aligned} & \beta_{k-1}^2(f_1(\mathbf{H}_k) - f_1(\mathbf{H}^*)) - \beta_k^2(f_1(\mathbf{H}_{k+1}) - f_1(\mathbf{H}^*)) \\ & \geq L\langle\beta_k(\mathbf{H}_{k+1} - \mathbf{Y}_k), \beta_k\mathbf{Y}_k - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*\rangle \\ & \quad + \frac{L}{2}||\beta_k\mathbf{H}_{k+1} - \beta_k\mathbf{Y}_k||_F^2 \\ & = \frac{L}{2}||\beta_k\mathbf{H}_{k+1} - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*||_F^2 \\ & \quad - \frac{L}{2}||\beta_k\mathbf{Y}_k - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*||_F^2 \\ & = \frac{L}{2}||\beta_k\mathbf{H}_{k+1} - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*||_F^2 \\ & \quad - \frac{L}{2}||(\beta_{k-1} - 1)(\mathbf{H}_k - \mathbf{H}_{k-1}) + \mathbf{H}_k - \mathbf{H}^*||_F^2 \\ & = \frac{L}{2}||\beta_k\mathbf{H}_{k+1} - (\beta_k - 1)\mathbf{H}_k - \mathbf{H}^*||_F^2 \\ & \quad - \frac{L}{2}||\beta_{k-1}\mathbf{H}_k - (\beta_{k-1} - 1)\mathbf{H}_{k-1} - \mathbf{H}^*||_F^2. \end{aligned} \quad (25)$$

Since $\beta_0 = 1$, then by varying the subscript in (25) from 1 to $k-1$, $(k \geq 2)$ and summing up all these inequalities, we have

$$\begin{aligned} & f_1(\mathbf{H}_1) - f_1(\mathbf{H}^*) - \beta_{k-1}^2(f_1(\mathbf{H}_k) - f_1(\mathbf{H}^*)) \\ & \geq \frac{L}{2}||\beta_{k-1}\mathbf{H}_k - (\beta_{k-1} - 1)\mathbf{H}_{k-1} - \mathbf{H}^*||_F^2 \\ & \quad - \frac{L}{2}||\mathbf{H}_1 - \mathbf{H}^*||_F^2 \\ & \geq -\frac{L}{2}||\mathbf{H}_1 - \mathbf{H}^*||_F^2. \end{aligned} \quad (26)$$

Regarding $f_1(\mathbf{H}_1) - f_1(\mathbf{H}^*)$ in the left of (26), we substitute $\mathbf{Y} = \mathbf{Y}_0, \hat{\mathbf{H}} = \mathbf{H}_1, \mathbf{H} = \mathbf{H}^*$ to (19) and get

$$\begin{aligned} & f_1(\mathbf{H}_1) - f_1(\mathbf{H}^*) \\ & \leq -\frac{L}{2}||\mathbf{Y}_0 - \mathbf{H}_1||_F^2 - L\langle\mathbf{H}_1 - \mathbf{Y}_0, \mathbf{Y}_0 - \mathbf{H}^*\rangle \\ & = \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}^*, 2\mathbf{Y}_0 - 2\mathbf{H}_1\rangle - \frac{L}{2}||\mathbf{Y}_0 - \mathbf{H}_1||_F^2 \\ & = \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}^*, \mathbf{Y}_0 - \mathbf{H}^* + \mathbf{H}^* - \mathbf{H}_1 + \mathbf{Y}_0 - \mathbf{H}_1\rangle \\ & \quad - \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}_1, \mathbf{Y}_0 - \mathbf{H}_1\rangle \\ & = \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}^*, \mathbf{Y}_0 - \mathbf{H}^*\rangle + \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}^*, \mathbf{H}^* - \mathbf{H}_1\rangle \\ & \quad + \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}^*, \mathbf{Y}_0 - \mathbf{H}_1\rangle - \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}_1, \mathbf{Y}_0 - \mathbf{H}_1\rangle \\ & = \frac{L}{2}||\mathbf{Y}_0 - \mathbf{H}^*||_F^2 + \frac{L}{2}\langle\mathbf{Y}_0 - \mathbf{H}^*, \mathbf{H}^* - \mathbf{H}_1\rangle \\ & \quad + \frac{L}{2}\langle\mathbf{H}_1 - \mathbf{H}^*, \mathbf{Y}_0 - \mathbf{H}_1\rangle \\ & = \frac{L}{2}(||\mathbf{Y}_0 - \mathbf{H}^*||_F^2 - ||\mathbf{H}_1 - \mathbf{H}^*||_F^2). \end{aligned} \quad (27)$$

Substituting (27) to (26), we have

$$\begin{aligned} & \beta_{k-1}^2(f_1(\mathbf{H}_k) - f_1(\mathbf{H}^*)) \\ & \leq \frac{L}{2}||\mathbf{H}_1 - \mathbf{H}^*||_F^2 + f_1(\mathbf{H}_1) - f_1(\mathbf{H}^*) \\ & \leq \frac{L}{2}||\mathbf{H}_1 - \mathbf{H}^*||_F^2 + \frac{L}{2}(||\mathbf{Y}_0 - \mathbf{H}^*||_F^2 - ||\mathbf{H}_1 - \mathbf{H}^*||_F^2) \\ & = \frac{L}{2}||\mathbf{Y}_0 - \mathbf{H}^*||_F^2. \end{aligned} \quad (28)$$

Since $\mathbf{Y}_0 = \mathbf{H}_0$ and $\beta_{k-1} \geq (k+1)/2$ based on [23], it holds from (28) that

$$\begin{aligned} f_1(\mathbf{H}_k) - f_1(\mathbf{H}^*) &\leq \frac{L}{2\beta_{k-1}^2}||\mathbf{H}_0 - \mathbf{H}^*||_F^2 \\ &\leq \frac{2L||\mathbf{H}_0 - \mathbf{H}^*||_F^2}{(k+1)^2}. \end{aligned} \quad (29)$$

This finishes the proof. ■

From the Proposition 1, one can see that the convergence rate in calculating the optimal $\mathbf{H}$ is $O(\frac{1}{k^2})$. Since the methods to update $\mathbf{H}$ and $\mathbf{W}$ are symmetric, we can conclude that our algorithm achieves the convergence rate $O(\frac{1}{k^2})$.

Regarding the computational complexity, the majority of time cost is spent on computing the gradient $\nabla_{\mathbf{H}} f_1(\mathbf{Y}_{k-1})$ (or $\nabla_{\mathbf{W}} f_2(\mathbf{X}_{k-1})$) within Step 2 (or Step 3). Note that $\nabla_{\mathbf{H}} f_1(\mathbf{Y}_{k-1})$ can be calculated by (4). Since $\mathbf{S}$, $\mathbf{W}$, and $\mathbf{V}$ are known before the calculation and the time cost on getting $\mathbf{S}^T\mathbf{W}^T$ is $O(mr^2)$, then the time cost on computing $\mathbf{S}^T\mathbf{W}^T\mathbf{W}\mathbf{S}$ and $\mathbf{S}^T\mathbf{W}^T\mathbf{V}$ in (4) is $O(2mr^2 + mrn)$. Therefore, the computational complexity of Step 2 is $O(2mr^2 + mrn) + K_1 \times O(nr^2)$. As for the computational complexity of Step 3, it is similar to that of Step 2 and can be denoted as $O(2mr^2 + mrn) + K_2 \times O(mr^2)$. Then, the total time cost of the proposed method in one iteration is $O(2mr^2 + 2mnr + 2nr^2) + K_1 \times O(nr^2) + K_2 \times O(mr^2)$. Since the iterative method for calculating $\mathbf{H}^*$ (or $\mathbf{W}^*$) achieves the nonlinear convergence rate $O(\frac{1}{k^2})$, the needed iteration numbers K_1 and K_2 are often small, typically less than r. As a result, our method performs much faster than many other methods with linear convergence rate.

III. SIMULATIONS

In this section, we use both computer generated data and real-world data to illustrate the performance of the proposed FNsNMF method, in terms of the CPU time t used, the factorization error e defined by

$$e = \frac{\|\mathbf{V} - \mathbf{WSH}\|_F^2}{\|\mathbf{V}\|_F^2} \tag{30}$$

and the factors' sparseness S measured by Hoyer's method in [17], i.e., for $\mathbf{V} \in \mathbf{R}_+^{m\times n}$,

$$S_{\mathbf{V}} = \frac{\sqrt{mn} - (\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{v}_{ij}) / \sqrt{\sum_{i=1}^{m}\sum_{j=1}^{n} \mathbf{v}_{ij}^2}}{\sqrt{mn} - 1}. \tag{31}$$

We also compare the performance of our method with that of the traditional NMF method [8], the NsNMF method [21], the NeNMF with L1-norm regularization (NENMF-L1R) [7], and the accelerated NMF method PGLINacc which also utilizes Lin's project gradient [11]. We would like to note that $\mathbf{S}$ in (30) should be the identity matrix for evaluating NMF, NeNMF-L1R, and PGLINacc. Each method is implemented using MATLAB R2011b installed in a personal computer with Intel(R) Core(TM) 2.5 GHz CPU, 8 GB memory and Microsoft Windows 8 operational system.

A. COMPUTER GENERATED DATA

In this subsection, the data matrix $\mathbf{V}$ is generated by multiplying two rank-50 matrices in $\Re^{100\times 50}$ and $\Re^{50\times 1000}$ (i.e., $m = 100, n = 1000$), respectively, whose entries are distributed uniformly within [0,1]. To get a highly faithful representation or small e index, the number of the features is set to be 50 (i.e., $r = 50$).

TABLE 1. Means of $e(*10^{-4})$, t, $S_{\mathbf{W}}$, and $S_{\mathbf{H}}$ obtained by different methods.

	FNsNMF	NMF	NsNMF	PGLINacc	NeNMF-L1R
e	0.9552	0.9981	0.9995	0.9422	0.9679
$S_{\mathbf{W}}$	0.2111	0.1433	0.1600	0.1367	0.2072
$S_{\mathbf{H}}$	0.3152	0.1449	0.1617	0.1290	0.2704
t	4.0578	73.0187	92.5891	16.0172	3.8176

We first test the time cost t and the sparseness degrees of the decompositions. For each compared algorithm, 100 runs are performed, with different initial values. For numerical comparisons, the results with $e < 10^{-4}$ and $\theta = 0.2$ are shown here. Table 1 gives the means of the indices e, t, $S_{\mathbf{W}}$, and $S_{\mathbf{H}}$, respectively. From this table, one can see that NsNMF and FNsNMF generate sparser decompositions than NMF and PGLINacc, due to the usage of non-smoothness constraint. Compared with NsNMF, our FNsNMF is much more efficient in computation.

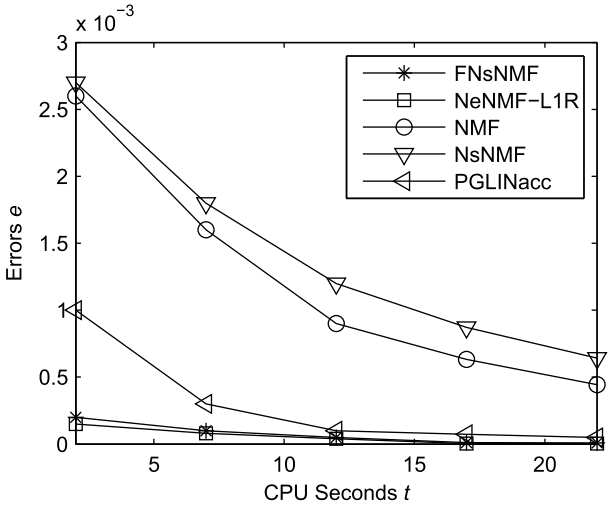

FIGURE 1. Error index e of each algorithm at different time instants.

Then, we compare the error indices of the above algorithms under different time costs. Fig.1 shows the averaged indices e of each algorithm based on 100 runs. One can see that our method is comparable to NeNMF-L1R, and it has smaller error than the remainder methods at each time instant. Specifically, it is much smaller than that of NsNMF which also utilizes non-smoothness constraint, meaning that the proposed FNsNMF speeds up NsNMF remarkably.

B. CBCL DATA

In this experiment, the real-world CBCL face data set is tested, which can be downloaded from [26]. It contains 2,429 19×19 facial low resolution gray-level images (i.e., $m = 381$ and $n = 2429$). Like that in [1], the feature number is set to be $r = 49$.

For each compared algorithm, 100 runs with different initial values are performed, and the mean results for reference

TABLE 2. Means of e, t, $S_{\mathbf{W}}$, and $S_{\mathbf{H}}$ obtained by different methods.

	FNsNMF	NMF	NsNMF	PGLINacc	NeNMF-L1R
e	0.0498	0.0499	0.0500	0.0428	0.0496
$S_{\mathbf{W}}$	0.6622	0.6119	0.6599	0.6020	0.5656
$S_{\mathbf{H}}$	0.5323	0.4986	0.5280	0.4637	0.5557
t	15.4531	44.9219	130.4609	19.4844	13.2614

are given in Table 2, where the errors are around 0.05 and $\theta = 0.2$. From this table, one can see that the averaged sparsity degrees of the decompositions obtained by FNsNMF and NsNMF are close. They are greater than those of NMF and PGLINacc, indicating the importance of the non-smoothness constraint in generating sparser results. Between NsNMF and the proposed FNsNMF, the latter is much faster than the former. It is worth noting that the mean of $S_{\mathbf{W}}$ and $S_{\mathbf{H}}$ of NeNMF-L1R is greater than that of NMF and PGLINacc, due to the L1-norm regularization. But it is smaller than that of FNsNMF and NsNMF.

C. COHN-KANADE DATA

In this simulation, we use the proposed method to learn sparse facial features and sparse coefficients to classify human face images, with comparisons to the related methods. This classification is beneficial to facial expression recognition. And we aim at the recognition of two universal expressions: happiness and surprise. The benchmark Cohn-Kanade data set (see also in [27]) released in 2000 is tested. This data set consists of 486 image sequences from 97 university students aged between 18 and 30 years, where 65% are female students, 15% are African-American, and 3% are from Asia or Latin American. Each image has 30×40 pixels, i.e., the dimensionality of the initial data is $m = 1200$. Fig. 2(a) shows 49 randomly observed images, where the first 25 images are with happiness expression and the remainder 24 images are with surprise expression.

In our experiments, the data dimensionality is reduced to $r = 49$ by using the compared NeNMF-L1R, NMF, NsNMF, PGLINacc and our FNsNMF, respectively. 50 random experiments are performed. The averaged errors for these methods are all around 0.02, and the averaged CPU-times are 5.3749, 26.3260, 44.3907, 13.9828, and 13.5236, respectively. We also calculate the sparseness degrees of the learned factors. The averaged sparseness degrees of the feature matrices are 0.3454, 0.4137, 0.4215, 0.3705, and 0.4417, respectively. Fig. 2(b-f) show the 49 features learned by these algorithms in a random experiment, respectively. As for the learned coefficient matrices, Fig. 3 shows the sparseness degrees in each experiment. It can be seen that our method generates the sparsest results. Notably that the sparseness is helpful to data storage, code, compression, and transfer. It is also beneficial for clustering and classification [28]. We will further show the effectiveness of the proposed method in classification.

Regarding the classification, we employ the popular support vector machine (SVM) method, and apply it to the low-dimensional data, i.e., the learned $\mathbf{H}$ in 50 experiments. The fivefold cross-validation scheme is utilized in the process of classification for each $\mathbf{H}$, i.e., each class data is partitioned into five complementary subsets, and in each iteration (of the total five), one subset is left for testing and the others are

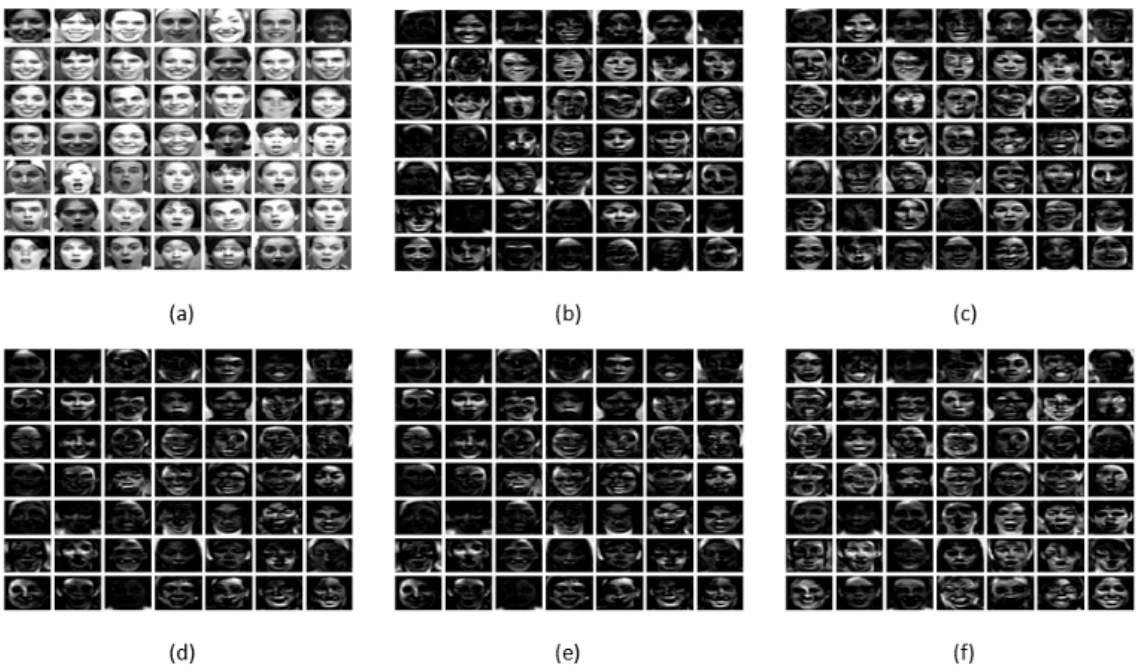

FIGURE 2. (a) The 49 initial happiness and surprise images, and the 49 basis images learned by (b) FNsNMF, (c) NeNMF-L1R, (d) NMF, (e) NsNMF, and (f) PGLINacc, respectively.

TABLE 3. Means of classification accuracy (%) of FNsNMF+SVM, NeNMF-L1R+SVM, NMF+SVM, NsNMF+SVM, and PGLINacc+SVM algorithms for the Cohn-Kanade database in 50 random experiments.

FNsNMF+SVM	NeNMF-L1R+SVM	NMF+SVM	NsNMF+SVM	PGLINacc+SVM
91.35%	90.95%	89.27%	90.46%	90.86%

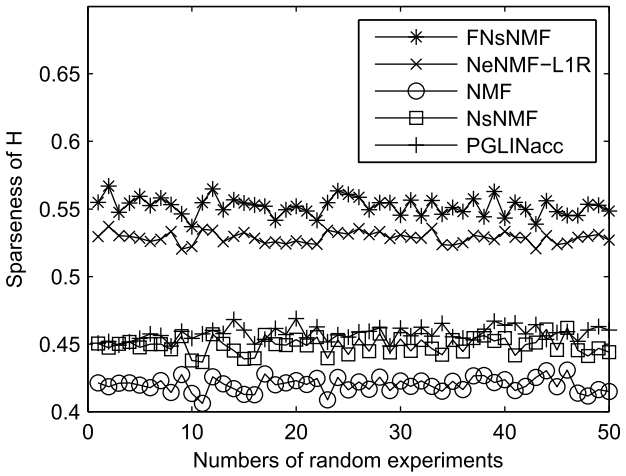

FIGURE 3. The sparseness indices of H learned in 50 random experiments by FNsNMF, NeNMF-L1R, NMF, NsNMF, and PGLINacc, respectively.

used for training. Finally, the classification accuracy of each compared algorithm is obtained by averaging the accuracies in each iteration. Table 3 gives the mean of the classification accuracies (%) in 50 experiments of these algorithms in combination with SVM. From this table, one can see that our method obtains the greatest accuracy index, implying that it is the most effective one among the compared methods.

IV. CONCLUSION

In this paper, a fast non-smooth NMF algorithm, called FNsNMF, is proposed for sparse data representation. Due to the usage of the non-smoothness factor, it obtains sparser decompositions than traditional NMF methods without constraint. Also, by using the proximal function and its highly efficient optimization scheme, the proposed method achieves a nonlinear convergence rate. As a result, it performs much faster than NsNMF which also employs the non-smoothness factor. Both computer generated data and real-world data are tested in the simulations, and the simulation results verify the advantages of our algorithm over the compared methods. In the future work, it is interesting to focus on how to extract common and individual features by using NMF-style methods for high efficient classification [29]. Also, it is meaningful to further analyze the stationarity of the compared algorithms. And this could be explored under the *block multi-convex structure* of the involved problem, which is proposed and discussed in [30].

REFERENCES

[1] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, pp. 788–791, Oct. 1999.

[2] R. Sandler and M. Lindenbaum, "Nonnegative matrix factorization with Earth mover's distance metric for image analysis," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 8, pp. 1590–1602, Aug. 2011.

[3] N. Guan, D. Tao, Z. Luo, and J. Shawe-Taylor. (2012). "MahNMF: Manhattan non-negative matrix factorization." [Online]. Available: http://arxiv.org/abs/1207.3438

[4] N. Guan, D. Tao, Z. Luo, and B. Yuan, "Online nonnegative matrix factorization with robust stochastic approximation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 23, no. 7, pp. 1087–1099, Jul. 2012.

[5] Z. Yang, Y. Xiang, K. Xie, and Y. Lai, "Adaptive method for nonsmooth nonnegative matrix factorization," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. [Online]. Available: http://dx.doi.org/, doi: 10.1109/TNNLS.2016.2517096.

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

[7] N. Guan, D. Tao, Z. Luo, and B. Yuan, "NeNMF: An optimal gradient method for nonnegative matrix factorization," *IEEE Trans. Signal Process.*, vol. 60, no. 6, pp. 2882–2898, Jun. 2012.

[8] D. D. Lee and H. S. Seung, "Algorithms for non-negative matrix factorization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 12. 2000, pp. 556–562.

[9] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 2, pp. 713–730, May 2008.

[10] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Comput.*, vol. 19, pp. 2756–2779, Oct. 2007.

[11] N. Gillis and F. Glineur, "Accelerated multiplicative updates and hierarchical ALS algorithms for nonnegative matrix factorization," *Neural Comput.*, vol. 24, no. 4, pp. 1085–1105, 2012.

[12] Z. He, S. Xie, R. Zdunek, G. Zhou, and A. Cichocki, "Symmetric nonnegative matrix factorization: Algorithms and applications to probabilistic clustering," *IEEE Trans. Neural Netw.*, vol. 22, no. 12, pp. 2117–2131, Dec. 2011.

[13] K. Huang, N. D. Sidiropoulos, and A. Swami, "Non-negative matrix factorization revisited: Uniqueness and algorithm for symmetric decomposition," *IEEE Trans. Signal Process.*, vol. 62, no. 1, pp. 211–224, Jun. 2014.

[14] N. Gillis and S. A. Vavasis, "Fast and robust recursive algorithms for separable nonnegative matrix factorization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 4, pp. 698–714, Apr. 2014.

[15] G. Zhou *et al.*, "Linked component analysis from matrices to high-order tensors: Applications to biomedical data," *Proc. IEEE*, vol. 104, no. 2, pp. 310–331, Feb. 2016.

[16] G. Zhou, A. Cichocki, Q. Zhao, and S. Xie, "Nonnegative matrix and tensor factorizations: An algorithmic perspective," *IEEE Signal Process. Mag.*, vol. 31, no. 3, pp. 54–65, May 2014.

[17] P. O. Hoyer, "Non-negative matrix factorization with sparseness constraints," *J. Mach. Learn. Res.*, vol. 5, pp. 1457–1469, Dec. 2004.

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

[19] H. Liu, Z. Wu, X. Li, D. Cai, and T. S. Huang, "Constrained nonnegative matrix factorization for image representation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 7, pp. 1299–1311, Jul. 2012.

[20] W. Ren, G. Li, D. Tu, and L. Jia, "Nonnegative matrix factorization with regularizations," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 4, no. 1, pp. 153–164, Mar. 2014.

[21] A. Pascual-Montano, J. M. Carazo, K. Kochi, D. Lehmann, and R. D. Pascual-Marqui, "Nonsmooth nonnegative matrix factorization (nsNMF)," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 3, pp. 403–415, Mar. 2006.

[22] Y. Nesterov, *Introductory Lectures on Convex Optimization: A Basic Course*. Boston, MA, USA: Kluwer, 2004.

[23] Y. Nesterov, "A method of solving a convex programming problem with convergence rate $O(1/k^2)$," *Sov. Math. Dokl.*, vol. 27, no. 2, pp. 372–376, 1983.

[24] Y. Nesterov, "Smooth minimization of non-smooth functions," *Math. Program.*, vol. 103, no. 1, pp. 127–152, 2005.
[25] R. Zdunek, "Alternating direction method for approximating smooth feature vectors in nonnegative matrix factorization," in *Proc. IEEE Int. Workshop Mach. Learn. Signal Process. (MLSP)*, Reims, France, Sep. 2014, pp. 1–6.
[26] (2005). *MIT Center for Biological and Computation Learning, CBCL Face Database 1*. [Online]. Available: http://www.ai.mit.edu/projects/cbcl
[27] T. Kanade, J. F. Cohn, and Y. Tian, "Comprehensive database for facial expression analysis," in *Proc. 4th IEEE Int. Conf. Autom. Face Gesture Recognit.*, Mar. 2000, pp. 46–53.
[28] D. Kuang, S. Yun, and H. Park, "SymNMF: Nonnegative low-rank approximation of a similarity matrix for graph clustering," *J. Global Optim.*, vol. 62, no. 3, pp. 545–574, Jul. 2015.
[29] G. Zhou, A. Cichocki, Y. Zhang, and D. P. Mandic, "Group component analysis for multiblock data: Common and individual feature extraction," *IEEE Trans. Neural Netw. Learn. Syst.*, to be published. [Online]. Available: http://dx.doi.org/10.1109/TNNLS.2015.2487364.
[30] Y. Xu and W. Yin, "A block coordinate descent method for regularized multiconvex optimization with applications to nonnegative tensor factorization and completion," *SIAM J. Imag. Sci.*, vol. 6, no. 3, pp. 1758–1789, 2013.

ZUYUAN YANG (M'15) received the B.E. degree from the University of Hunan University of Science and Technology in 2003, Xiangtan, China, and the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2010. He won the excellent Ph.D. Thesis Award Nomination of China. He joined the National Program for New Century Excellent Talents in University and won the Guangdong Distinguished Young Scholar. He is currently a Researcher with the School of Automation, Guangdong University of Technology, Guangzhou, China. His research interests include machine learning, nonnegative matrix factorization, and image processing.

YU ZHANG was born in Hubei, China. He received the B.E. degree from the School of Automation, Guangdong University of Technology, Guangzhou, China, in 2015. He is currently pursuing the master's degree with the School of Automation, Guangdong University of Technology, Guangzhou, China. His research interests include machine learning and nonnegative matrix factorization.

WEI YAN was born in Jiangxi, China. He received the master's degree from the School of Automation, Guangdong University of Technology, Guangzhou, China, in 2016. He is currently pursuing the Ph.D. degree with the Faculty of Science and Technology, University of Macau. His research interests include machine learning, nonnegative signal processing, and blind signal processing.

YONG XIANG (SM'12) received the Ph.D. degree in electrical and electronic engineering from The University of Melbourne, Australia. He is currently a Professor and the Director of the Artificial Intelligence and Image Processing Research Cluster, School of Information Technology, Deakin University, Australia. His research interests include signal and system estimation, information and network security, multimedia (speech/image/video) processing, and wireless sensor networks. He has authored over 130 refereed journal and conference papers in these areas. He is an Associate Editor of the IEEE Signal Processing Letters and the IEEE Access. He has served as a Program Chair, a TPC Chair, a Symposium Chair, and a Session Chair for a number of international conferences.

SHENGLI XIE (M'0–SM'02) was born in Hubei, China, in 1958. He received the M.S. degree in mathematics from Central China Normal University, Wuhan, China, and the Ph.D. degree in control theory and applications from the South China University of Technology, Guangzhou, China, in 1992 and 1997, respectively. He is currently the Director of the Laboratory for Intelligent Information Processing and a Full Professor with the School of Automation, Guangdong University of Technology, Guangzhou. He has authored or co-authored two monographs, a dozen of patents, and more than 80 papers in journals and conference proceedings. His current research interests include automatic controls, signal processing, blind signal processing, and image processing.