Journal of Computational Mathematics
Vol.30, No.1, 2012, 34–46.

http://www.global-sci.org/jcm
doi:10.4208/jcm.1110-m11si10

AN EFFECTIVE INITIALIZATION FOR ORTHOGONAL NONNEGATIVE MATRIX FACTORIZATION*

Xuansheng Wang
School of Mathematical Science, Xiamen University, Xiamen 361005, China
Email: wxs111111@163.com
Xiaoyao Xie
School of Mathematics and Computer Science, Guizhou Normal University, Guiyang 550001, China
Email: xyx@gznu.edu.cn
Linzhang Lu
School of Mathematics and Computer Science, Guizhou Normal University, Guiyang 550001, China
School of Mathematical Science, Xiamen University, Xiamen 361005, China
Email: lzlu@xmu.edu.cn

Abstract

The orthogonal nonnegative matrix factorization (ONMF) has many applications in a variety of areas such as data mining, information processing and pattern recognition. In this paper, we propose a novel initialization method for the ONMF based on the Lanczos bidiagonalization and the nonnegative approximation of rank one matrix. Numerical experiments are given to show that our initialization strategy is effective and efficient.

Mathematics subject classification: 65F99.
Key words: Lanczos bidiagonalization, Orthogonal nonnegative matrix factorization, Low-rank approximation, Nonnegative approximation.

1. Introduction

Let m and n be two integers, denote by $\mathbb{R}_+^{m,n}$ the set of all $m \times n$ nonnegative matrices. The nonnegative matrix factorization (NMF) problem means that for given $A \in \mathbb{R}_+^{m,n}$ and $k \ll \min(m,n)$, finding $W \in \mathbb{R}_+^{m,k}$ and $H \in \mathbb{R}_+^{k,n}$ such that

$$A \approx WH. \tag{1.1}$$

That is, finding two nonnegative matrices of low rank W and H, such that their product is an approximation of a given nonnegative matrix A in some distance metrics (in this paper, the distance metric will be the Frobenius norm $\|.\|_F$) [14]. The NMF, or approximation of a nonnegative matrix, has become a useful tool in a large applications, such as, images processing, text mining and space situation alertness. Scientific literature and soft tools [4] on the subject and variants thereof are rapidly expending. The orthogonal nonnegative matrix factorization (ONMF), where an orthogonality constraint is imposed on a factor (W or H) in the decomposition (1.1), was shown to provide a more clear interpretation on a link between clustering and matrix decomposition [5]. Multiplicative updates for the NMF with preserving orthogonality were recently proposed in [3]. Numerical experiments on face image data for an image representation task show that the ONMF algorithm preserves the orthogonality, while

* Received February 28, 2011 / Revised version received June 7, 2011 / Accepted June 20, 2011 / Published online January 9, 2012 /

the goodness-of-fit (GOF) is minimized. In [3], the GOF is compared with standard NMF. As this is not the point of this paper, we will not describe it in details.

To speed up the convergence of the NMF methods and the minimization of the objective function, most research papers to date for the NMF algorithms have discussed the need to investigate good initialization strategies [1]. However, few of them mentioned the initialization of the ONMF. Therefore, in this paper, we propose a novel initialization algorithm for the ONMF based on the Lanczos algorithm and nonnegative approximation of rank one matrices (see [2]). The proposed algorithm has some good features: it can be combined with all ONMF algorithms and allows a little randomization by free choice of the initial vectors in the Lanczos process. Moreover, our initialization can preserve some original information from given data. From our numerical experiments, it is seen that the initialization algorithm work effectively and efficiently.

The rest of this paper is organized as follows. Section 2 reviews the Lanczos bidiagonalization process to get a low-rank approximation of a nonnegative matrix. Section 3 presents and analyzes our algorithm. In section 4, we give some numerical experiments to demonstrate our algorithms. The last section provides some conclusion.

2. Lanczos Bidiagonalization

Since the ONMF is a constrained low-rank approximation problem of a matrix, we need to seek an initialization strategy among alternative low-rank factorizations. For such a problem, the following Eckart-Young theorem [12] is important.

Theorem 2.1. *Let $A \in \mathbb{R}^{m,n}$ have the singular values decomposition (SVD)*

$$A = P\Sigma Q^T, \quad \Sigma = diag(\sigma_1, \sigma_2, \cdots, \sigma_n) \in \mathbb{R}^{m,n},$$

where $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n \geq 0$ are the singular values *of A, $P \in \mathbb{R}^{m,m}$ and $Q \in \mathbb{R}^{n,n}$ are orthogonal matrices. Then for $1 \leq r \leq n$, the matrix*

$$A_r = Pdiag(\sigma_1, \cdots, \sigma_r, \underbrace{0, \cdots, 0}_{n-r})Q^T \tag{2.1}$$

is a global minimize of the optimization problem

$$\min \left\{ \|A - B\|_F^2 \ \ | B \in \mathbb{R}^{m,n}, rank(B) \leq r \right\} \tag{2.2}$$

with the corresponding minimum value $\sum_{i=r+1}^{n} \sigma_i^2$. Moreover, if $\sigma_r > \sigma_{r+1}$, then A_r is the unique global minimizer.

It follows from Theorem 1 that once the SVD of a matrix A is available, the best rank r approximation A_r of A is easily computed. When A is large, however, computing the SVD of A can be costly. If we are only interested in some A_r with $r \ll \min(m, n)$, the computation of the complete SVD of A is rather wasteful. It is therefore desirable to develop less expensive alternatives for computing a good approximation of A_r. In this section, we show that we can obtain a good low-rank nonnegative approximation of a nonnegative matrix A directly from the Lanczos bidiagonalization process without computing the SVD of A.

In the following, we describe the Lanczos bidiagonalization process presented in [11, 20].

Let b be a starting vector, for $i = 1, 2, \cdots, k$, compute

$$\beta_1 u_1 = b, \;\; \alpha_1 v_1 = A^T u_1, \;\; \beta_{i+1} u_{i+1} = A v_i - \alpha_i u_i, \;\; \alpha_{i+1} v_{i+1} = A^T u_{i+1} - \beta_{i+1} v_i. \tag{2.3}$$

In (2.3), α_i and β_i should be chosen such that $\|u_i\|_2 = \|v_i\|_2 = 1$, so they can be positive or negative. However, for our application (see next section), we shall choose α_i and β_i to be positive.

In compact matrix form (2.3) can be written as

$$U_{k+1}(\beta_1 e_1) = b, \quad A V_k = U_{k+1} B_{k+1}(:, 1:k), \quad A^T U_k = V_{k+1} B_{k+1}^T, \tag{2.4}$$

where $B_{k+1} \in \mathbb{R}^{(k+1),(k+1)}$ is lower bidiagonal,

$$B_{k+1} = \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_{k+1} & \alpha_{k+1} \end{bmatrix}, \qquad \begin{aligned} U_{k+1} &= [u_1, u_2, \cdots, u_{k+1}], \\ V_{k+1} &= [v_1, v_2, \cdots, v_{k+1}], \end{aligned}$$

and $B_{k+1}(:, 1:k)$ is B_{k+1} with the last column deleted.

If we hope to find a few dominant singular value triplets of A, we must compute the SVD of B_k in (2.4). The singular values of B_k are used as approximations of the singular values of A, and the singular vectors of B_k are combined with left and right Lanczos vectors $\{u_i\}_1^k$ and $\{v_i\}_1^k$ to form approximations of the singular vectors of A [9].

As we are only interested in finding a low-rank approximation of A, it is expected that a cheaper and more direct approach is used without computing the SVD of B_k. So it is quite natural to choose

$$J_k = U_k B_k V_k^T \tag{2.5}$$

as a rank-k approximation of A. It is obvious that if $\alpha_i > 0$, $i = 1, \cdots, k$, then $rank(J_k) = k$. Let A_j be defined in (2.1). It was showen in [20] that for any $k > j$, $||A - J_k||_F^2$ will approach $||A - A_j||_F^2$ when k gets large. Furthermore, [20] gave many examples that illustrate even for a k that is only slightly larger than j, $||A - J_k||_F^2$ is already very close to $||A - A_j||_F^2$.

Following the error analysis in [18], it is straightforward to show that in finite precision arithmetic, (2.4) become

$$\hat{U}_{k+1}(\hat{\beta}_1 e_1) = b, \;\; \hat{A}\hat{V}_k = \hat{U}_{k+1}\hat{B}_{k+1}(:, 1:k) + F_k, \;\; \hat{A}^T\hat{U}_{k+1} = \hat{V}_{k+1}\hat{B}_{k+1}^T + G_{k+1}, \tag{2.6}$$

where $\|F_k\| = \mathcal{O}(\|A\|_F \epsilon_M)$ and $\|G_{k+1}\| = \mathcal{O}(\|A\|_F \epsilon_M)$ with ϵ_M the machine epsilon, and

$$F_i = [f_1, f_2, \cdots, f_i], \quad G_i = [g_1, g_2, \cdots, g_i].$$

In (2.6), by adding "$\hat{}$", we denote the computed version of a quantity.

In [20], the authors further discussed stopping criterion (ω_k=$\|A - J_k\|_F \leq tol$, tol is a user supplied tolerance) of the Lanczos process and reorthogonalization process to maintain an adequate level of orthogonality (see [20]). But, for our application, once the low rank k is given, the orthogonality is not so important.

The following Lanczos bidiagonalization process is that our initialization method described in next section needs.

Algorithm 2.1 Lanczos bidiagonalization process in Matlab notation
Inputs: Matrix $A \in R_+^{m,n}$, positive vector $b \in R_+^m$, integer $0 < k < \min(m,n)$.
Outputs: Rank-k matrix $U \in R^{m,k}$, $B \in R_+^{k,k}$, $V \in R^{k,n}$.
1. $\beta = norm(b);\ u = b/\beta;\ v = A^T u;\ \alpha = norm(v);\ v = v/\alpha;\ B(1,1) = \alpha;$
2. $U = [u];\ V = [v];$
3. **for** $j = 2 : k$
$u = A * v - \alpha * u;\ \beta = norm(u);\ u = u/\beta;\ B(j, j-1) = \beta;\ U = [U, u];$
$v = A^T * u - \beta * v;\ \alpha = norm(v);\ v = v/\alpha;\ B(j,j) = \alpha;\ V = [V, v].$
end

3. Initialization of (W, H)

For convenience, we denote by $A \geq B$ the componentwise inequality $\alpha_{i,j} \geq \beta_{i,j}$ for all elements of (equisized matrices) A, B. Given any vector or matrix variable X, its "positive section", $X_+ \geq 0$, will be defined to be the vector or matrix of the same size that contains the same values as X where X has nonnegative elements and 0 elsewhere. The "negative section" of X will be the matrix $X_- = X_+ - X$, where again $X_- \geq 0$. It follows that any vector or matrix X can be written as $X = X_+ - X_-$, and if $X \geq 0$ then $X_- = 0$.

Based on the above Lanczos bidiagonalization process, in this section we present a method for initialization of (W, H) in the decomposition (1.1) that turns out to be quite effective. Our strategy is similar to the strategy used in [2], so we first review some results given in [2].

Lemma 3.1. *For any matrix $C \in \mathbb{R}^{m,n}$ of rank one, $rank(C_+) \leq 2$, $rank(C_-) \leq 2$.*

Lemma 3.2. *Assume $C \in \mathbb{R}^{m,n}$ has unit rank, so that $C = uv^T$ for some $u \in \mathbb{R}^m$, $v \in \mathbb{R}^n$. Let also $\hat{u}_\pm := u_\pm/\|u_\pm\|, \hat{v}_\pm := v_\pm/\|v_\pm\|$ be the normalized positive and negative sections of u and v, and $\sigma_\pm = \|u_\pm\|\|v_\pm\|$ and $\eta_\pm = \|u_\pm\|\|v_\mp\|$. Then the unordered singular value expansions of C_+ and C_- are*

$$C_+ = \sigma_+ \hat{u}_+ \hat{v}_+^T + \sigma_- \hat{u}_- \hat{v}_-^T, \qquad C_- = \eta_+ \hat{u}_+ \hat{v}_-^T + \eta_- \hat{u}_- \hat{v}_+^T. \tag{3.1}$$

Furthermore, the maximum singular triplet of C_+ is $(\sigma_+, \hat{u}_+, \hat{v}_+)$ if

$$\sigma_+ = \max\Big(\|u_+\|\|v_+\|, \|u_-\|\|v_-\|\Big);$$

otherwise it is $(\sigma_-, \hat{u}_-, \hat{v}_-)$. Similarly, the maximum singular triplet of C_- is $(\eta_+, \hat{u}_+, \hat{v}_-)$ if

$$\eta_+ = \max\Big(\|u_+\|\|v_-\|, \|u_-\|\|v_+\|\Big);$$

otherwise it is $(\eta_-, \hat{u}_-, \hat{v}_+)$.

Lemma 3.3. *Given $C \in \mathbb{R}^{m,n}$ having unit rank. If C contains both positive and negative elements, then $rank(C_+) = rank(C_-) = 2$. If $C \geq 0$ (resp. $C \leq 0$) then $rank(C_+) = 1$ (resp. $rank(C_-) = 1$).*

Lemma 3.4. *If $C \in \mathbb{R}^{m,n}$ satisfying $rank(C) = 1$, then $C_+ = \arg\min_{G \in \mathbb{R}_+^{m,n}} \|C - G\|_F$.*

From these lemmas, we know that the best (in terms of the Frobenius norm) nonnegative approximation of the unit rank matrix $C = uv^T$ would be C_+.

Now we derive our initialization algorithm. From now on, we assume that the given matrix A is nonnegative. In Algorithm 2.1, we always take a positive vector b as the starting vector and set $\alpha_i > 0,\ \beta_i > 0$. Since A is nonnegative, the vectors u_1 and v_1 are nonnegative. However, this can not guarantee that J_k of (2.5) produced from Algorithm 2.1 is nonnegative. So we has to use a modification of J_k that will finally produce a nonnegative approximation of the nonnegative matrix A. For this, rewrite J_k as

$$\begin{aligned} J_k = U_k B_k V_k^T &= [u_1, u_2, \cdots, u_k] \begin{bmatrix} \alpha_1 & & & \\ \beta_2 & \alpha_2 & & \\ & \ddots & \ddots & \\ & & \beta_k & \alpha_k \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}, \\ &= [u_1, \cdots, u_k] \begin{bmatrix} \alpha_1 & & & \\ & \alpha_2 & & \\ & & \ddots & \\ & & & \alpha_k \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix} + [u_1, \cdots, u_k] \begin{bmatrix} 0 & & & \\ \beta_2 & 0 & & \\ & \ddots & \ddots & \\ & & \beta_k & 0 \end{bmatrix} \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_k \end{bmatrix}, \\ &= \sum_{i=1}^{k} \alpha_i C_i + \sum_{i=1}^{k-1} \beta_{i+1} D_i, \end{aligned} \tag{3.2}$$

where $C_i = u_i v_i^T$ and $D_i = u_{i+1} v_i^T$ are rank one matrices.

Let $C_i = C_{i+} - C_{i-}$ and $D_j = D_{j+} - D_{j-}$. Define $\{\sigma_j(C_{i+}), m_j(C_{i+}), n_j(C_{i+})\}$ and $\{\mu_l(D_{j+}), x_l(D_{j+}), y_l(D_{j+})\}$ as the singular triplets of C_{i+} and D_{j+} by nonincreasing order, respectively. Then we have

$$\begin{aligned} J_k &= \sum_{i=1}^{k} \alpha_i u_i v_i^T + \sum_{i=1}^{k-1} \beta_{i+1} u_{i+1} v_i^T, \\ &= \sum_{i=1}^{k} \alpha_i C_i + \sum_{i=1}^{k-1} \beta_{i+1} D_i, \\ &= \alpha_1 C_1 + \sum_{i=2}^{k} \alpha_i C_{i+} - \sum_{i=2}^{k} \alpha_i C_{i-} + \sum_{i=1}^{k-1} \beta_{i+1} D_{i+} - \sum_{i=1}^{k-1} \beta_{i+1} D_{i-}, \\ &= \alpha_1 C_1 + \sum_{i=2}^{k} \alpha_i \sigma_1(C_{i+}) m_1(C_{i+}) n_1(C_{i+})^T + \sum_{i=1}^{k-1} \beta_{i+1} \mu_1(D_{i+}) x_1(D_{i+}) y_1(D_{i+})^T + E, \end{aligned}$$

where

$$E = \sum_{i=2}^{k} \alpha_i \Big(\sigma_2(C_{i+}) m_2(C_{i+}) n_2(C_{i+})^T - C_{i-} \Big) + \sum_{i=1}^{k-1} \beta_{i+1} \Big(\mu_2(D_{i+}) x_2(D_{i+}) y_2(D_{i+})^T - D_{i-} \Big).$$

Thus, we can choose initial value of (W, H) such that

$$\begin{aligned} WH &= \alpha_1 C_1 + \sum_{i=2}^{k} \alpha_i \sigma_1(C_{i+}) m_1(C_{i+}) n_1(C_{i+})^T + \sum_{i=1}^{k-1} \beta_{i+1} \mu_1(D_{i+}) x_1(D_{i+}) y_1(D_{i+})^T \\ &= J_k - E. \end{aligned} \tag{3.3}$$

Define $\hat{E} = A - J_k = A - (WH + E)$, $R = A - WH = \hat{E} - E$. and set $\omega_k = \|A - J_k\|_F < tol$ (tol is a user supplied tolerance). We have

Proposition. Given $A \in \mathbb{R}_+^{m,n}$, and suppose that the pair (W, H) is initialized by (3.3). Then the Frobenius norm of $R = A - WH$ is bounded by

$$\|E\|_F \leq \|R\|_F \leq \|\hat{E}\|_F + \|E\|_F \leq tol + \|E\|_F.$$

Though the upper bounds are loose, it indicates that the residual is bounded. Of far greater interest is that, in practice, by applying our initial strategy only few iterations can drive the residual down to a magnitude that is very close to the one we would have obtained has we applied the underlying ONMF algorithms with random initialization but for many more iterations.

The two major computational steps of our initial strategy are

(i) running k-step of Lanczos bidiagonalization process to get the matrix J_k of (2.5),

(ii) computing the nonnegative approximation matrices (W, H) from (3.3).

Note that from Lemma 3.2, it is not necessary for us to do really SVD to $\{C_{i+}\}$ and $\{D_{i+}\}$. Thus, from the above analysis, we can deduce the overall cost for our initialization strategy on dense matrix A is $\mathcal{O}(kmn)$.

The preceding results constitute the theoretical foundation of our initialization method, which can be implemented as in the following algorithm.

Algorithm 3.1. Lanczos based initialization algorithm in Matlab notation.
Inputs: Matrix $A \in R_+^{m,n}$, positive vector b, integer $0 < k < \min(m, n)$.
Outputs: Rank-k nonnegative matrices $W \in R_+^{m,k}$ and $H \in R_+^{k,n}$.

1. Run **Algorithm 2.1** to get rank-k matrices $U \in R^{m,k}$, $V \in R^{k,n}$.
2. Initialize $W(:,1) = U(:,1); H(1,:) = V(:,1)'$;
for $i = 2 : k$
3. $uu = U(:,i); vv = V(:,i); vv1 = V(:,i-1)$;
4. $uup = pos(uu); uun = neg(uu); vvp = pos(vv); vvn = neg(vv)$;
5. $vvp1 = pos(vv1); vvn1 = neg(vv1)$;
6. $nuup = norm(uup); nvvp = norm(vvp); nuun = norm(uun)$;
7. $nvvn = norm(vvn); nvvp1 = norm(vvp1); nvvn1 = norm(vvn1)$;
8. $termp = nuup * nvvp; termn = nuun * nvvn$;
9. $termp1 = nuup * nvvp1; termn1 = nuun * nvvn1$;
10. **if** (termp>=termn) $W(:,i) = B(i,i) * uup/nuup$; $H(i,:) = vvp'/nvvp$;
else $W(:,i) = uun/nuun$; $H(i,:) = vvn'/nvvn$;
end
11. **if** (termp1>=termn1) $W(:,i) = W(:,i) + uup/nuup$;
$H(i,:) = H(i,:) + vvp1'/nvvp1$;
else $W(:,i) = W(:,i) + uun/nuun$; $H(i,:) = H(i,:) + vvn1'/nvvn1$.
end
end

In Algorithm 3.1, functions **pos** and **neg** extract the positive and negative sections of their argument. $[A_p] = pos(A)$ returns $A_p = (A >= 0). * A$, and $[A_n] = neg(A)$ returns

$A_n = (A < 0). * (-A)$.

The ONMF is the NMF with orthogonality on either the factor W or H [6, 7] in the decomposition (1.1). For example, the objective functions for one-side ONMF with respect to the factor W is symbolically written as

$$F = \min \parallel A - WH \parallel_F^2, \;\; s.t \;\; W^T W = I, \; W \geq 0, \; H \geq 0, \tag{3.4}$$

where I is the identity matrix.

By introducing the Lagrangian multiplier, the approximate solution to the constrained optimization problem (3.4) can be found. The multiplicative update rules for (3.4) are computed as follows [7]:

$$W = W. * ((AH^T)./(WWTAHT))^\eta, \tag{3.5}$$

$$H = H. * (W^T A)./(WTWH). \tag{3.6}$$

Where η is a constant, $.*$ and $./$ denote elementwise multiplication and division. Thus we obtain the ONMF algorithm which constraints orthogonality of the factor W, which will be called the ONMFW_η algorithm.

ONMFW_η algorithm (ONMF algorithm with factor W orthogonally constrained)
Inputs: Matrix $A \in R_+^{m,n}$, positive vector $b \in R_+^m$, integer $0 < k < \min(m, n)$ and coefficient η.
Outputs: Rank-k nonnegative matrices $W \in R_+^{m,k}$ and $H \in R_+^{k,n}$.
1. Choose randomly rank-k nonnegative matrices $W \in R_+^{m,k}$ and $H \in R_+^{k,n}$;
2. Update W and H with update rule (3.5) and (3.6) until convergence.

Similarly, the objective functions for one-side ONMF with respect to the factor H is symbolically written as

$$F = \min \parallel A - WH \parallel_F^2, \;\; s.t \;\; HH^T = I, \; W \geq 0, \; H \geq 0. \tag{3.7}$$

The multiplicative update rules for (3.7) are

$$W = W. * (AH^T)./(WHHT), \tag{3.8}$$

$$H = H. * ((W^T A)./(WTAHTH))^\eta. \tag{3.9}$$

Also, we can obtain the ONMF algorithm which constraints orthogonality of the factor H, which is called the ONMFH_η algorithm.

ONMFH_η algorithm (ONMF algorithm with factor H orthogonally constrained)
Inputs: Matrix $A \in R_+^{m,n}$, positive vector $b \in R_+^m$, integer $0 < k < \min(m, n)$ and coefficient η.
Outputs: Rank-k nonnegative matrices $W \in R_+^{m,k}$ and $H \in R_+^{k,n}$.
1. Choose randomly rank-k nonnegative matrices $W \in R_+^{m,k}$ and $H \in R_+^{k,n}$;
2. Update W and H with update rule (3.8) and (3.9) until convergence.

By combining the initialization strategy (algorithm 3.1) described in previous section with the update rules (3.5), (3.6), we can deduce an algorithm for the ONMF which constraints orthogonality of the factor W. This algorithm is called the LanIN ONMF W_η algorithm.

Algorithm 3.2. (**LanIN ONMF W_η algorithm**).
Inputs: Matrix $A \in R_+^{m,n}$, positive vector $b \in R_+^m$, integer $0 < k < \min(m,n)$ and coefficient η.
Outputs: Rank-k nonnegative matrices $W \in R_+^{m,k}$ and $H \in R_+^{k,n}$.
1. Run **Algorithm 2.1 and Algorithm 3.1** to get rank-k matrices $B \in R_+^{k,k}$, $W \in R_+^{m,k}$, $H \in R_+^{k,n}$.
2. Initialize $d = norm(W(:,1)); D = [d]; W(:,1) = W(:,1)/d;$
for i=2: k
$d = norm(W(:,i)); W(:,i) = W(:,i)/d;$
$D = [D,d];$
end
$H = diag(D) * B * H;$
3. Update W and H with update rule (3.5) and (3.6) until convergence.

Similar to Algorithm 3.2, we deduce an algorithm in which the factor H is constrained to be orthogonal, and we call this algorithm the LanIN ONMF H_η algorithm.

Algorithm 3.3. (**LanIN ONMF H_η algorithm**).
Inputs: Matrix $A \in R_+^{m,n}$, positive vector $b \in R_+^m$, integer $0 < k < \min(m,n)$ and coefficient η.
Outputs: Rank-k nonnegative matrices $W \in R_+^{m,k}$ and $H \in R_+^{k,n}$.
1. Run **Algorithm 2.1 and Algorithm 3.1** to get rank-k matrices $B \in R_+^{k,k}$, $W \in R_+^{m,k}$, $H \in R_+^{k,n}$.
2. Initialize $c = norm(H(1,:)); C = [c]; H(1,:) = H(1,:)/c;$
for i=2: k
$c = norm(H(i,:)); H(i,:) = H(i,:)/c;$
$C = [C,c];$
end
$W = W * B * diag(C);$
3. Update W and H with updating rules (3.8) and (3.9) until convergence.

4. Numerical Experiments

From previous sections, we can see that our initialization method can readily be combined with all existing ONMF algorithms. Moreover, our initialization method may have a little random since the starting vector b can be randomly chosen and multiple runs are allowed.

Table 4.1: Example 2: Comparison the objective value and orthogonality between our strategy and the original algorithms without initialization.

	k	iterations	objective value	orthogonality
ONMF$W_{0.5}$	10	600	149.9801	0.6848
LanIN ONMF$W_{0.5}$	10	600	147.9692	0.6718
ONMFW_1	5	200	105.5465	0.6193
LanIN ONMFW_1	5	200	105.4855	0.5129

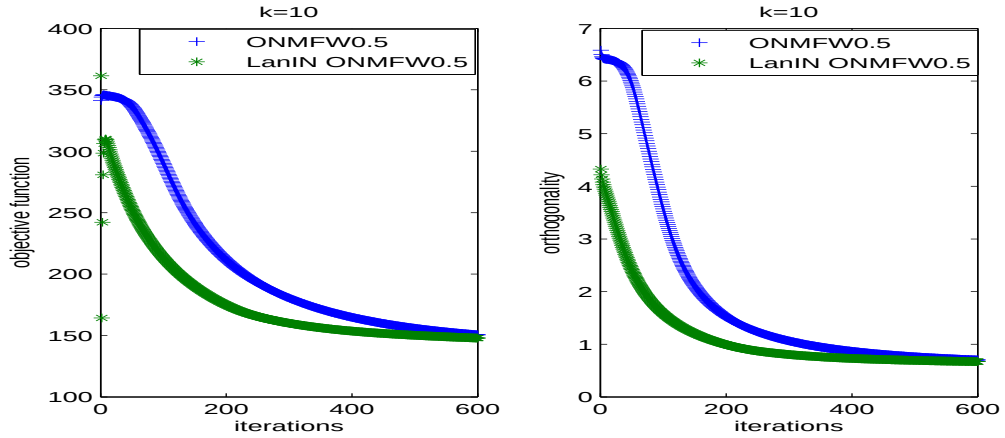

Fig. 4.1. Example 1: Comparison of objective function (left) and orthogonality of W (right) for CBCL images between LanIN ONMF$W_{0.5}$ algorithm and ONMF$W_{0.5}$ algorithm.

By choosing different starting vector $b > 0$ in Lanczos process, our algorithms may produce different (W, H). However, in the following experiments, we always take $b = (1, 1, \cdots, 1)^T$. The objective function, orthogonality measure for W and H are chosen as $(||A - W * H||_F)$, $(||eye(k) - W^T * W||_2)$ and $(||eye(k) - H^T * H||_2)$, respectively. All computation is done using Matlab version 7 on an Gennuine Intel(R) CPU @1.86G HZ, 1.5 EMS memory computer. In all experiments, we fill in the zeros with random values in the space: [0:average/100].

Taking the coefficient $\eta = 0.5$ and $\eta = 1$, we do several experiments to compare the ONMF algorithms with the initialization (LanIN ONMFW_η and LanIN ONMFH_η) with those without the initialization (ONMFW_η and ONMFH_η).

Example 1. Consider the CBCL images (http://cbcl.mit.edu/ cbcl/software-datasets/FaceData2.html), which is a 361×2429 matrix. Figs. 4.1 and 4.2 depict the experiment results with the LanIN ONMF$W_{0.5}$ algorithm and ONMF$W_{0.5}$ algorithm. The parameter (rank) k is 10 and iteration number is 600. Naturally, we select $W = rand(361, k)$, $H = rand(k, 2429)$ as initial pair (W, H) in ONMF$W_{0.5}$ algorithm.

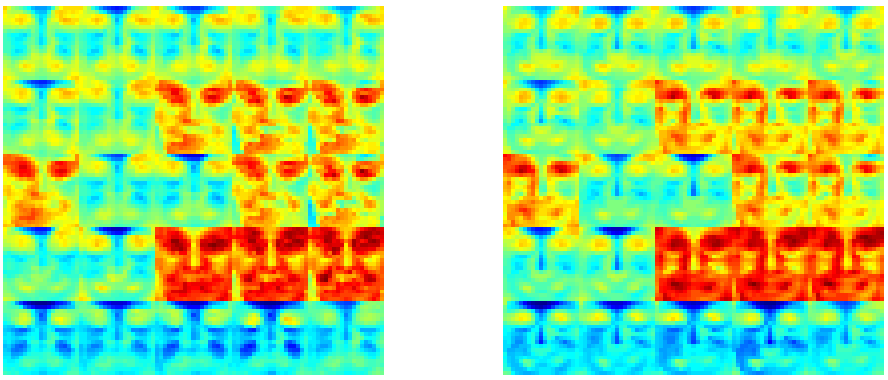

Fig. 4.2. Example 1: Approximation on CBCL images using ONMF$W_{0.5}$ algorithm (left) and LanIN ONMF$W_{0.5}$ algorithm (right).

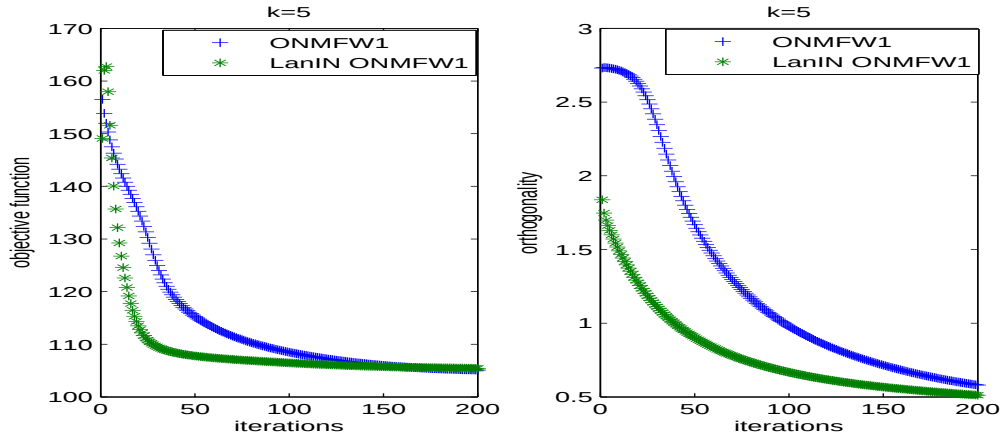

Fig. 4.3. Example 2: Comparison of objective function (left) and orthogonality of W (right) for CBCL images between LanIN ONMFW_1 algorithm and ONMFW_1 algorithm.

Example 2. We use the same data set as in Example 1, and we compare the ONMFW_1 algorithm with LanIN ONMFW_1 algorithm. The parameter (rank) k is 5 and the iteration number is 200. We select $W = rand(361, k)$, $H = rand(k, 2429)$ as initial pair (W, H) in the ONMFW_1 algorithm. Figs. 4.3 and 4.4 present the results.

It is observed from Figs. 4.1–4.4 and Table 4.1 that the objective function value and the

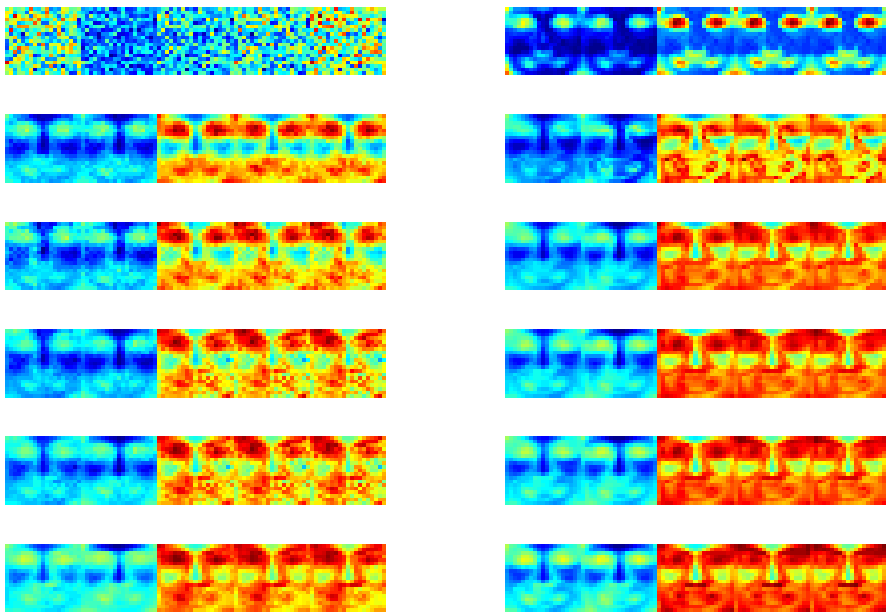

Fig 4.4. Example 2: Progress of approximations on CBCL images (from 16th to 20th images) using: ONMFW_1 algorithm (left) and LanIN ONMFW_1 algorithm (right). Row correspond to 0, 20, 50, 80, 100 and 200 iterations using $k = 5$.

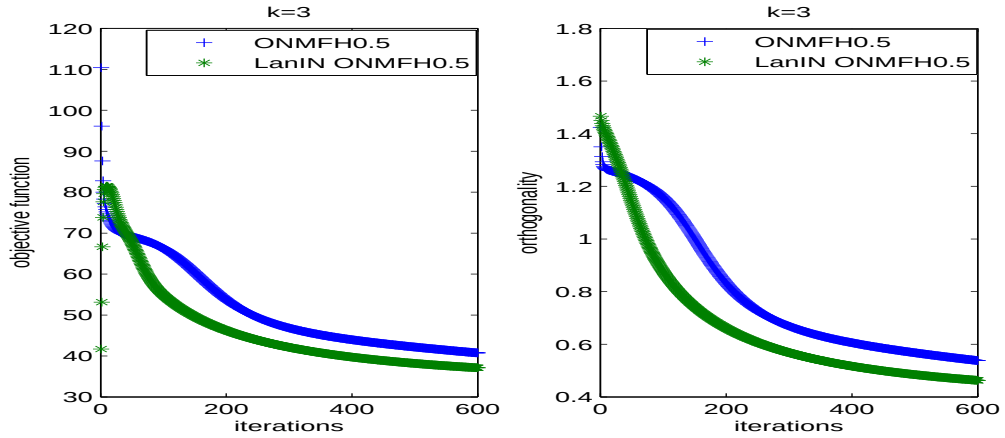

Fig 4.5. Example 3: Comparison of objective function (left) and orthogonality of H (right) for Kuls images between LanIN ONMF$H_{0.5}$ algorithm and ONMF$H_{0.5}$ algorithm.

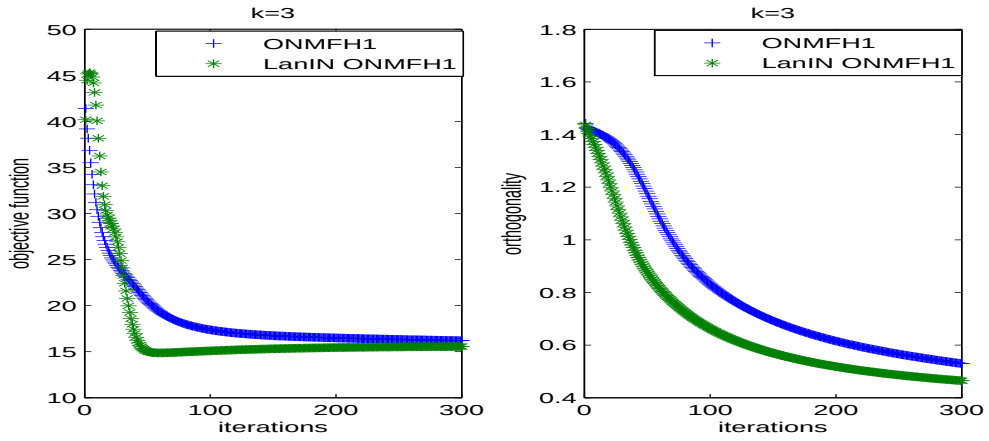

Fig 4.6. Example 4: Comparison of objective function (left) and orthogonality of H (right) for Kuls images between LanIN ONMFH_1 algorithm and ONMFH_1 algorithm.

orthogonality using initialization can converge much faster than that without the initialization. The approximation images of the LanIN ONMF$W_{0.5}$ and LanIN ONMFW_1 also are seen to contain more original image information than those of the ONMF$W_{0.5}$ and ONMFW_1.

Example 3. In the third example, the data set is Kuls illuminated faces in [15] (it is a 4096×20 matrix). Fig. 4.5 depicts the results with LanIN ONMF$H_{0.5}$ algorithm and ONMF$H_{0.5}$ algorithm. The parameter (rank) k is 3 and iteration number is 600. Also, we select $W = rand(4096, k)$, $H = rand(k, 20)$ as initial pair (W, H) in ONMF$H_{0.5}$ algorithm.

Example 4. In this example, we use same data set as in Example 3. The difference here is that we compare the ONMFH_1 algorithm with LanIN ONMFH_1 algorithm. The parameter (rank) k is 3 and the iteration number is 300. We select $W = rand(4096, k)$, $H = rand(k, 20)$ as initial pair (W, H) in the ONMFH_1 algorithm. Fig. 4.6 presents the results of the two algorithms.

It is seen from above experiments that aided with our initialization method, the objective function values and the orthogonality can converge much faster and the approximation images

Fig 4.7. Example 4: Left: original Kuls images; center: approximation on Kuls images using ONMFH_1 algorithm; right: approximation on Kuls images using LanIN ONMFH_1 algorithm.

can also get more original information from the original images. It implies that our initialization strategy works effectively and efficiently.

5. Conclusion

Based on the Lanczos bidiagonalization process and nonnegative approximation of rank-one matrix, we derive an effective initialization algorithm for the orthogonal nonnegative factorization of a nonnegative matrix. This algorithm can readily be combined with the existing ONMF algorithms and contains a little random because of free choice of the starting vector b. It is seen from the numerical experiments that our initialization strategy combines with the existing ONMF algorithms can converge much faster and works better than the ONMF algorithms without the initialization.

It is easy to know from the above discussion that the initialization strategy can also apply to other nonnegative matrix factorization, for example, the orthogonal nonnegative trifactorizations proposed in [5]. It is a decomposition with orthogonality on both of the factors W and H. Similarly, the initialization method is applicable when a nonnegative symmetric A needs a nonnegative symmetric decomposition.

Acknowledgments. The work is supported by National Natural Science Foundation of China No. 10961010.

References

[1] M.W. Berry, M. Browne, A.N. Langvile, V.P. Pauca, R.J. Plemmons, Algorithms and applications for approximation nonnegative matrix factorization, *Comput. Stat. Data An.*, **52** (2007), 115-173.

[2] C. Boutsidis, E. Gallopoulos, SVD based initialization: A head start of nonnegative matrix factorization, *Pattern Recogn.*, **41** (2008), 1350-1362.

[3] S. Choi, Algorithms for orthogonal nonnegative matrix factorization. In Proceedings of the International Joint Conference on Neural Networks (IJCNN), Hong Kong, 2008.

[4] A. Cichocki, R. Zdunek, NMFLAB MATLAB toolbox for vsugunar@yahoo.co.in nonnegative matrix factorization. `URL: http://www.bsp.brain.riken.jp/ICALAB/nmflab.html/`.

[5] C. Ding, T. Li, W. Peng, H. Park, orhtogonal nonnegative matrix factorization for clustering. In proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2006.
[6] C. Ding, X. He, H.D. Simon, On the equivalence of nonnegative matrix factoriation and spectral clustering, In: SIAM Data Mining Conf, 2005.
[7] C. Ding, T. Li, W. Peng, H. Park, Orthogonal nonnegative matrix t-factorizations for clustering. In: KDD '06: Proc. 12th ACM SIGKDD Internat. Conf. on Knowledge Discovery and Data Mining, ACM, 2006, 126-135.
[8] J. Cohen, U. Rothblum, Nonnegative ranks, decomposition, and factorizations of nonnegative matrices, *Linear Algerbra Appl.*, **190** (1993), 149-168.
[9] J. Cullum, R.A. Willoughby, and M. Lake, A Lanczos algorithm for computing the singular values and corrsponding singular vectors of a matrix, *ACM T. Math. Software*, **7** (1981), 140-169.
[10] L. Elden, Matrix Methods in Data Mining and Pattern Recognition, SIAM, Philadelphia, PA, USA, 2007.
[11] G.H. Gloub and C.F. Van Loan, Matrix Computation, 3rd ed., The Johns Hopkins University Press, Baltimore, MD. 1996.
[12] N.J. Higham, Matrix nearest problems and applications, in: M. Gover, S. Barnett (Eds.), Applications of Matrix Theory, Oxford University Press, 1989, pp.1-27.
[13] D.D. Lee, H.S. Seung, Unsupervised learning by convex and coinc coding, *Adv. Neural Inf. Process. Syst.*, **9** (1997) 515-521.
[14] D.D. Lee, H. S. Seung, Algorithms for non-negative matrix factorization, *Adv. Neural Inf. Process. Syst.*, **13** (2001).
[15] Nicolas Gillis, Francois Glineur Using Underapproximations for Sparse Nonnegative Matrix Factorization, *Pattern Recogn.*, (2010), 1676-1687.
[16] P. Paatero, Least squares formulation of robust non-negative factor analysis, *Chemometr. Intell. lab.*, **37** (1997), 23-35.
[17] P. Paatero, U. Tapper, Positive matrix factorization: a non-negative factor model with optimal utilization of error estiamtes of data values, *Environmetrics*, **5** (1994), 111-126.
[18] C.C. Paige, Errors analysis of the Lanczos algorithm for tridiagonalizaing a symmetric matrix, *J. Inst. Math. Appl.*, **18** (1976), 341-349.
[19] R. Salakhutdinov, S. Roweis, Z. Ghahramani, On the convergence of bound optimization algorithms, in: Proceedings of the 19th Conference in Uncertainty in Artifical Intelligence (UAI'03), Morgan Kanfamann, Los Altos, CA, 2003, 509-516.
[20] H.D. Simon and H. Zha, Low-rank Matrix Approximation Using Lanczos Bidiagonalization Process with Applications, *SIAM J. Sci. Comput.*, **21**:6 (2000), 2257-2274.
[21] Jiho Yoo, and Seungjin Choi, Orthogonal nonnegative matrix tri-factorization for co-clustering: Multiplicative updates on Stiefel manifolds, *Inform. Process. Manag.*, **46** (2010), 559-570.