A Non-Convex Optimization Framework for Large-Scale Low-rank Matrix Factorization

SUBMISSION DATE / POSTED DATE

27-04-2020 / 20-07-2021

CITATION

Hafshejani, Sajad Fathi; Vahidian, Saeed; Moaberfard, Zahra; Lin, Bill (2020): A Non-Convex Optimization Framework for Large-Scale Low-rank Matrix Factorization. TechRxiv. Preprint. https://doi.org/10.36227/techrxiv.12199026.v2

DOI

10.36227/techrxiv.12199026.v2

A Non-Convex Optimization Framework for Large-Scale Low-rank Matrix Factorization

Sajad Fathi Hafshejani[2], Saeed Vahidian[1], Zahra Moaberfard[2], and Bill Lin[1]

[1] Dept. of Electrical and Computer Engineering
University of California San Diego, CA, USA.

[2] Dept. of Mathematics
Shiraz University of Technology, Shiraz, Iran

Abstract

Low-rank matrix factorization problems such as non negative matrix factorization (NMF) can be categorized as a clustering or dimension reduction technique. The latter denotes techniques designed to find representations of some high dimensional dataset in a lower dimensional manifold without a significant loss of information. If such a representation exists, the features ought to contain the most relevant features of the dataset. Many linear dimensionality reduction techniques can be formulated as a matrix factorization. In this paper, we combine the conjugate gradient (CG) method with the Barzilai and Borwein (BB) gradient method, and propose a BB scaling CG method for NMF problems. The new method does not require to compute and store matrices associated with Hessian of the objective functions. Moreover, adopting a suitable BB step size along with a proper nonmonotone strategy which comes by the size of convex parameter η_k, results in a new algorithm that can significantly improve the CPU time, efficiency, the number of function evaluation. Convergence result is established and numerical comparisons of methods on both synthetic and real-world datasets show that the proposed method is efficient in comparison with existing methods and demonstrate the superiority of our algorithms.

1 Introduction

Matrix completion and factorization algorithms that do not require non-negativity on their components, such as principal component analysis (PCA) or independent component analysis (ICA), often do not provide interpretable decompositions. In contrast, non-negative matrix factorization (NMF) is a method that promotes relatively spatial representation and has gained a lot of attention

recently within both the computer science and optimization communities due to its applications in data science problems [1]. By imposing non-negative structures simultaneously in data reconstruction and basis identification, NMF has shown great potential in unravelling important structures in the data from a lot of applications, including data clustering [4], protein sequence motif discovery [3], and etc. NMF is based on the assumption that a $m \times n$ data matrix (dataset) X with n data and m features, can be represented by a small sets of k basis vectors [2]. A non-negative data matrix X (m features by n data) into two non-negative sub-matrices W and H, such that

$$X \approx \{WH \quad \Big| \; X \in \mathbb{R}_+^{m \times n}, W \in \mathbb{R}_+^{m \times k}, H \in \mathbb{R}_+^{k \times n}\} \tag{1}$$

The rank, k typically is much smaller than the dimensions and data ($k \ll m, n$). There exist many different methods to solve the NMF problem. Since (1) is an approximation problem, one should look for a suitable cost function to measure the goodness of fit. The most common loss function used in the literature is the squared Frobenius norm loss:

$$\min_{W,H \geq 0} f(W,H) = \frac{1}{2}\|X - WH\|_F^2. \tag{2}$$

While other loss functions are available, using the Frobenius norm is well-motivated due to its compatibility with the Singular Value Decomposition (SVD). By defining the cost function as in (2), the next step would be choosing the optimization method. So far, different algorithms have been suggested. Early proposed methods mainly concentrated on multiplicative update rules. For instance, [2] proposed the following algorithm.

$$W_{ia}^{k+1} = W_{ia}^{k} \frac{(XH^{k^T})_{ia}}{(W^k H^k H^{k^T})_{ia}}, \qquad \forall i, a; \tag{3}$$

$$H_{bj}^{k+1} = H_{bj}^{k} \frac{(W^{k+1^T} X)_{bj}}{(W^{k+1^T} W^{k+1} H^k)_{bj}}, \qquad \forall b, j. \tag{4}$$

This method updates W and H individually, such that the Frobenius norm loss in NMF above is guaranteed to decrease on each iterations. However, since multiple matrix products need to be calculated on each iterations, potential computation savings might be available by modifying the algorithm. More variations of multiplicative update rules can be seen in [5]. Since the NMF is not jointly convex in W and H, many other optimization algorithms has been proposed, such as block-coordinate descent, projected gradient descent, and non-negative least squares (ANLS). In particular, ANLS modifies the problem in (2) into two convex optimization problems:

$$W^{k+1} = \min_{W \geq 0} f(W, H^k) = \min_{W \geq 0} \frac{1}{2}\|V - WH^K\|_F^2, \tag{5}$$

$$H^{k+1} = \min_{H \geq 0} f(W^{k+1}, H) = \min_{H \geq 0} \frac{1}{2}\|V - W^{k+1}H\|_F^2 \tag{6}$$

which can be solved using standard convex optimization solvers. An important characteristic of this method is that, since the constraints are built directly into the subproblems, the limit points of any algorithm that solves the subproblems will also be stationary points for the NMF problem [6]. The proposed algorithms for solving this problem is quite diverse. For instance, one method, known as the active set method [6] and an additional method, Hierarchical Alternating Least Squares (HALS), simplifies the nonnegative subproblem by updating each column of W and row of H individually [7]. On a similar note, much attention has been put in prior works to solve these problems [8] by employing variants of steepest descent algorithm for unconstrained problems. In [9, 10], the so-called Rank-one Residue Iteration (RRI) algorithm for solving the NMF problem was studied. This algorithm was independently proposed by Cichocki, Zdunek, and Amari in [11], where it is named the Hierarchical Alternating Least Squares (HALS) algorithm. A helpful feature of this algorithm is that the solution to (5), and (6) have explicit formulas which make it easy to implement. Simulation experiments in [10, 11] show that the HALS/RRI algorithm is much faster than the Lee–Seung method and also some other methods based on ANLS methods. Using Newton or quasi–Newton methods to solve the subproblems can have a faster rate of convergence [12]. However, these methods require to determine a suitable active set for the constraints at each iteration [13]. Another approach for solving the NMF problem is to apply the CG method. This strategy has very fast and has global convergence.

In this paper, motivated by the success of the prior works of projected gradient method and good performance of the BB method for optimization, we propose a novel non-monotone CG method that uses BB method to overcome the aforementioned problems for NMF. By considering that the objective function of each NMF subproblem is a convex function whose gradient is Lipchitz continuous, we optimally minimize one matrix factor with another fixed. In particular, we update two sequences recursively for optimizing each factor, the step size is determined by the BB method, and the search point is constructed by CG method. This algorithm accelerates the optimization based on the problem structure. Therefore, this method converges much faster than other NMF solvers by alternatively performing the optimal gradient method on the two matrix factors.

Preliminary experiments on both synthetic and real-world datasets suggest the efficiency of the proposed method. Face recognition experiments on popular real-world and synthetic datasets confirm the effectiveness of PNBB. In addition, we show that PNBB can be naturally extended to handle several versions of NMF that incorporates varients of BB step sizes.

1.1 BB Step Size and Nonmonotone Algorithm

To begin, in this paper we deal with solving the following minimization problem as:

$$\min_{x \in \mathbb{R}^n_+} f(x)$$

where $f(x)$ stands for $f(W, H)$ for short. There are various optimization algorithms for solving this problem. Indeed, they compute the search direction and find a suitable step size and move to next point. One of the most popular choices for the search directions is generated by the Newton

method or the quasi-Newton method [14]. Although these methods have superlinear convergence, they are not suitable for solving large-scale optimization problems, since it is difficult to compute and store the (approximate) Hessian matrices of function at each iteration [15]. To overcome the drawbacks in the Newton-type or the quasi-Newton-type methods, numerous spectral gradient methods [16] and CG methods [15] have been presented so far. Numerical experiments are often employed to show the advantages of their numerical efficiency in solving large-scale optimization problems. With that in mind, herein we make use of CG method. In particular, at kth iteration the new point is computed as

$$x_{k+1} = x_k + \alpha_k d_k, \tag{7}$$

where α_k and d_k are named the step size and search direction respectively. The search direction is obtained by using the following strategy:

$$d_{k+1} = \begin{cases} g_1 = -g_0 & \text{if } k = 1 \\ -g_{k+1} + \beta_k d_k & \text{if } k > 1. \end{cases} \tag{8}$$

where $\beta_k = \frac{g_k^{\mathsf{T}}(g_k^{\mathsf{T}} - g_{k-1})}{\|g_{k-1}\|^2}$ and $g_k = \nabla f(x_k)$.

To find α_k, the traditional monotone line search approaches depart from x_k along the search direction d_k such that $f(x_k + \alpha d_k) < f(x_k)$. In exact line search method, the step size is obtained by solving the following minimization problem:

$$\phi(\alpha) = \min_{\alpha > 0} \phi(x_k + \alpha d_k). \tag{9}$$

According to [17], this strategy is expensive and impractical when the cost of the minimization problem with multiple variable required. On the other hand, inexact methods such as Armijo condition, the Wolfe condition and the Goldstein condition are other alternative methods for finding an acceptable step size α_k. For instance, the Armijo line search finds the step size α_k such that

$$f(x_{k+1}) < f(x_k) + \alpha_k \gamma \nabla f(x_k)^T d_k, \tag{10}$$

where $\gamma \in (0, 1)$ is a constant. By using the fact that d_k is a descent direction i.e. $g_k^T d_k < 0$, we can conclude that the traditional Armijo rule guarantees the monotonicity of the sequence $\{f_k\}$. While all the above-mentioned algorithms are monotone, Chamberlain et al. [18] suggested an interesting nonmonotone line search technique namely "watchdog technique" for improving the iterative algorithms. Since then, the nonmonotone technique has been exploited by many researchers. In detail, their method finds a step size α satisfying the following condition:

$$f(x_k + \alpha_k d_k) \leq f_{l_k} + \gamma \alpha_k g_k^T d_k, \qquad k \geq 0, \tag{11}$$

where

$$\begin{gathered} f_{l_k} = \max_{0 \leq j \leq n_k} \{f_{k-j}\}, \\ n_0 = 0, \;\; 0 \leq n_k \leq \min\{n_{k-1} + 1, N\} \end{gathered} \tag{12}$$

where $N \geq 0$ is a fixed number and f_{l_k} is the so-called non-monotone term and allows the objective function to increase in some iterations when its convergence is guaranteed. Employing the nonmonotone strategy with other approaches such as Newton's method, trust region method and CG method has led to a significant improvement [19]. Later on, Grippo and Sciandrone in [20] proposed a non-monotone line search technique for Newton's method. However, Grippo's non-monotone technique has some drawbacks [21], for example, the iterative method may generate R-linear convergent iterations for strongly convex function, the iterations may not satisfy the condition (11) for k sufficiently large, for any fixed bound N on the memory. Also, in some cases, the performance of the algorithm highly depend on the choice of N. Further, in their proposed method, the algorithm takes a constant value as the step length in the first iteration and cannot be adjusted according to the characteristics of the objective function in the current iteration.

Zhang and Hager [22] proved that the best convergence results are always obtained by using a stronger non monotone strategy when iterations are far from the minimizer point, and by using a weaker non monotone strategy when iterations are close to it. In this regard, Ahookhosh et al. [21] suggested a new non-monotone condition as:

$$f(x_k + \alpha_k d_k) \leq R_k + \gamma \alpha_k g_k^T d_k, \tag{13}$$

where R_k is defined by

$$\begin{gathered} R_k = \eta_k f_{l_k} + (1 - \eta_k) f_k, \\ \eta_k \in [\eta_{\min}, \eta_{\max}],\ \eta_{\min} \in [0, 1),\ \eta_{\max} \in [\eta_{\min}, 1]. \end{gathered} \tag{14}$$

The drawback of (14) is that the value of step size in initial steps of inner loop is one and close to one which increases the number of iterations of the inner loop. To cure this issue, we apply a new and efficient Barzilai-Borwein (BB) which is motivated by Newton's method but is Hessian free at almost no extra cost over the standard gradient method. Also, the algorithm often significantly outperform the standard gradient method. In particular, for computing the BB step size, we need to define $s_k = x_k - x_{k-1}, y_k = g_k - g_{k-1}$ based on which we are able to present the BB step sizes as follows

$$\alpha^1 = \frac{s_k^T y_k}{y_k^T y_k} \qquad and \quad \alpha^2 = \frac{s_k^T s_k}{s_k^T y_k}.$$

Inspired by the BB methods step sizes, an extended BB step size which is obtained by $\sqrt{\alpha^1 \alpha^2}$ is also employed in this paper. In the next section we will discuss the algorithm in detail.

2 The Optimization Algorithm

Here in this section, we will focus on the proposed algorithm for solving problems (5) and (6). As mentioned in [19], to obtain the best convergence results, we need a stronger non-monotone strategy whenever the iterates are far from the minimizer and a weaker non-monotone strategy for those iterates that are close enough to that. To this end, here we introduce a new parameter η_k that

can address the mentioned drawbacks in the preceding section. The parameter η_k is estimated by the algorithm at each iteration as bellow:

$$\eta_{k+1} = 1 - \exp\left(-\|g_k\|\right). \tag{15}$$

It is clear that in the first iterations that are far away from the minimizer, the values of parameter η_k are closed to 1. On the other hand, when the $\|g_k\|$ goes to zero, the values of the parameter η_k goes to zero. Much faster convergence is usually observed in the above-mentioned algorithm compared to its peer, although the cost function does not decline significantly at initial stages which is due to way that η_k is defined in their formulations. To be more concrete, η_k should be defined such that to be contingent upon the behavior of the gradient of the cost function. Therefore, it is advantageous to design monotone parameters to be close to one in initial stages of the algorithm and vanishes at the final iterations. This will be the underpinning motivation of our algorithm design.

One of the other parameters that can influence the performance of the algorithm is N which was defined in the nonmonotone term in (12). It should be noted that N is assumed to be fixed in almost all non-monotone optimization algorithms. In this paper, we consider an adaptive value of N which depends on the value of the gradient at each step. The update of N in each iteration is done as follows

$$N_{k+1} = \begin{cases} N_k + 1, & 10^{-1} < \|g_k\|; \\ N_k, & 10^{-3} \leq \|g_k\| \leq 10^{-1}; \\ N_k - 1, & otherwise. \end{cases} \tag{16}$$

By taking (16) into account, when the sequence of iterations is located in a narrow valley, that is, the norm of gradient is large, to avoid creeping along the bottom of a narrow curved valley, it is better that we increase the value of N_k. Besides, when the norm of gradient is moderate and satisfies the second condition, it would be better not to change the value of the N_k. In the last case, where the norm of gradient is small that is the current iteration is in a flat area, to further decrease the function value, it is better for the algorithm to shrink the value of N_k.

Now, we in position to focus on designation of the algorithm for updating one of these matrices (W and H) and the same process can be applied to update the other matrix. Let the algorithm update the matrix W. As such, we deal with solving the following problem:

$$\min_{W \geq 0} f^k(W) := f(W, H^k) = \frac{1}{2}\|V - WH^k\|_F^2, \tag{17}$$

For simplicity, we use $f(W) := f^k(W)$ for short. It is clear that $f(W)$ is a convex function, therefore the optimal point can be found.

Now, in order to obtain the new point, we first need to define a quadratic form of the objective function $f(W)$. For this purpose, we consider the quadratic form of the function ϕ for any fixed $U \geq 0$ as:

$$\phi(U, W) := f(U) + \langle \nabla f(U), W - U \rangle + \frac{1}{2}\|W - U\|_F^2.$$

Algorithm 1 Algorithm of PNBB

Initialization and Input $0 < \alpha_{\min} \leq \alpha_{\max}, \eta_0 \in (0,1], \gamma \in (0,1), W_0 \in \mathbb{R}^{m \times k}, \rho = 0.25,\ N_0 = 5\ k \in \{1, ..., \text{stopping criteria}\}$
While $(\|P[W_k - \nabla f(W_k)] - W_k\| \geq \varepsilon$
    Generate a descent direction D_k
    Compute the new point
    While Eq. (13) is False
        $\alpha \leftarrow \rho\alpha$
    end
    $\alpha_k \leftarrow \alpha$
    $W_{k+1} \leftarrow W_k + \alpha_k D_k$
    Compute the α_{k+1} by (21)
    Calculate η_{k+1} by using (15)
    Calculate N_{k+1} by using (16)
    $k \leftarrow k + 1$
end
return W

It is obvious that the function $\phi(U, W)$ is strictly convex in term W for any fixed $U \geq 0$. Therefore, the optimal solution for this function can be obtained. At the current iteration j, a new point is generated by solving the strongly convex quadratic minimization problem as:

$$\min_{W \geq 0} \phi(W_k, W). \tag{18}$$

Now, by using some calculations, we can derive a unique solution for (18) as:

$$D_k = -\nabla f(W) + \beta D_{k-1}. \tag{19}$$

Therefore, the new point is constructed as:

$$W_{j+1} = P[W_k + \alpha_k D_k], \tag{20}$$

where all the negative entries of X are projected to zero by using the projection operator $P[.]$. Finally, by using the BB method, the step size α_k is computed by:

$$\alpha_{k+1} \leftarrow \min\left\{\alpha_{\max}, \max\left\{\alpha_{\min}, \alpha^3\right\}\right\} \tag{21}$$

where $\alpha^1 = \frac{\langle S_{k-1}, Y_{k-1}\rangle}{\langle Y_{k-1}, Y_{k-1}\rangle}$, $\alpha^2 = \frac{\langle S_{k-1}, S_{k-1}\rangle}{\langle S_{k-1}, Y_{k-1}\rangle}$, and $\alpha^3 = \sqrt{\alpha^1 \alpha^2}$.

We outlined the mentioned procedure in Algorithm 1 named as projected nonmonotone BB (PNBB) algorithm.

3 Convergence Analysis

The aim of this section is two folds: first, we prove that the proposed nonmonotone strategy is a descent method which can be proved according to the structure of the nonmonotone equation. Second, we prove the convergence of the algorithm. In doing so, we obtain a lower bound for the step size and then will demonstrate that for enough number of iterations, the norm of the search direction goes to zero implying that the convergence occurs. To begin, we establish two assumptions thereby we will investigate the convergence analysis. Afterwards, we delineate some properties of the new proposed non-monotone term which leads us to prove that the sequence W_t generated by the Algorithm 1 converges to a stationary point of (5). The so-called assumptions are as follows:

A1: The level sets, i.e., the sequence $\{W_k\}$ generated by Algorithm 1 is contained in the level set

$$L(W) = \{W | f(W) \leq f(W_0), \quad W \geq 0\}.$$

A2: The gradient of this function, that is:

$$\nabla f(W) = (WH^k - V)(H^k)^T \tag{22}$$

is Lipschitz continuous with constant $L = \|H^k(H^k)^T\|_F$.

Next, we elaborate on some lemmas that delineates some properties of the new non-monotone line search proposed in this paper.

Lemma 1. Let $\{W_k\}$ be the sequence obtained by Algorithm 1, then, $f_{l(k)}$ is a diminishing sequence.

Proof. The proof comes by taking into account the definition of R_k and (12), which renders that

$$R_k = \eta_k f_{l(k)} + (1 - \eta_k) f_k \leq \eta_k f_{l(k)} + (1 - \eta_k) f_{l(k)} = f_{l(k)}. \tag{23}$$

It can be followed by

$$f_{k+1} \leq R_k + \lambda\gamma\langle\nabla f(Z_k), D_k\rangle \leq f_{l(k)} + \lambda\gamma\langle\nabla f(Z_k), D_k\rangle.$$

Due to the negativity of $\langle\nabla f(W_k), D_k\rangle < 0$, we can conclude that

$$f_{k+1} \leq f_{l(k)}. \tag{24}$$

Besides, from (12) we have the following result.

$$\begin{aligned} f_{l(k+1)} &= \max_{0\leq j\leq m(k+1)} \{f_{k+1-j}\} \\ &\leq \max_{0\leq j\leq m(k)+1} \{f_{k+1-j}\} = \max\left\{f_{l(k)}, f_{k+1}\right\}. \end{aligned} \tag{25}$$

From (24) and (25) we reach $f_{l(k+1)} \leq f_{l(k)}$ which demonstrate that the sequence $f_{l(k)}$ is a diminishing sequence. □

Lemma 2. For the sequence $\{W_k\}$ generated by Algorithm 1 and for all $k \geq 0$, we have $W_k \in L(W_0)$.

Proof. By the definition of $f_{l(k+1)}$ in mind, we can posit $f_{k+1} \leq f_{l(k+1)}$ for any $k \geq 0$. Thus, we have:

$$\begin{aligned} f_{k+1} &= \eta_{k+1} f_{k+1} + (1-\eta_{k+1}) f_{k+1} \leq \eta_{k+1} f_{l(k+1)} \\ &\quad + (1-\eta_{k+1}) f_{k+1} = R_{k+1}, \forall\, k \in N_k \end{aligned} \tag{26}$$

Now, by making use of the definition of R_k, we can conclude that $R_0 = f_0$. Next, by induction, assuming $W_i \in L(W_0)$, for all $i = 1, 2, \ldots, k$, we show that $W_{k+1} \in L(W_0)$. Equations (12) and (23) along with Lemma 1 is followed by $f_{k+1} \leq f_{l(k+1)} \leq f_{l(k)} \leq f_0$, which implies that the sequence W_k is contained in $L(W_0)$. □

In order to establish the convergence, Karush-Kuhn-Tucker (KKT) optimality conditions require to hold for problem (17)

$$W \geq 0;\ \nabla f(W) \geq 0;\ \nabla f(W) \otimes W = 0,$$

where $\otimes$ stands for the Hadamard product. Further, prior to talking about the stationary points, we need to define the scaled projected gradient direction as follows.

$$D_\alpha = -\nabla f(W) + \beta D, \tag{27}$$

where $\alpha > 0$ and $W \geq 0$. Now, we are in position to investigate the stationary point and some facts about scaled projected gradient direction by the following lemmas.

Lemma 3. Let $\alpha \in (0, \alpha_{\max}]$ and $W \geq 0$. Thus, we have:

- $\langle \nabla f(W), D_\alpha(W) \rangle \leq -\frac{1}{\alpha}\|D_\alpha\|^2 \leq -\frac{1}{\alpha_{\max}}\|D_\alpha\|^2$
- W is a stationary point of (17) if and only if $D_\alpha(W) = 0$.

Proof. The proof is provided in [26]. □

Lemma 4. Assuming the first part of Lemma (3) and (H1) hold and Algorithm 1 produces the sequence $\{W_k\}$, the sequence $\{f_{l(k)}\}$ will be convergent.

Proof. Lemma 1 along with the fact that $f_{l(0)} = f_0$ it will be straightforward to conclude that the sequence $\{W_{l(k)}\}$ remains in level set $L(W_0)$. In addition, $f(W_k) \leq f(W_{l(k)})$ shows that the sequence $\{W_k\}$ remains in $L(W_0)$. Thus, (A1) along with Lemma 1 imply that the sequence $\{f_{l(k)}\}$ is convergent. □

Lemma 5. With (A1) and that the direction D_k satisfies the first item of Lemma 1 in mind, for the sequence $\{W_k\}$ generated by Algorithm 1, we get $\lim_{k\to\infty} f_{l(k)} = \lim_{k\to\infty} f_k$.

Proof. The proof can be done in the same fashion as in Lemma 2 of [21]. □

Lemma 6. Assume that (A1) holds and the direction D_k satisfies the first item of Lemma 1. Then, for the sequence $\{W_k\}$ generated by Algorithm 1, we can show

$$\lim_{k\to\infty} R_k = \lim_{k\to\infty} f_k.$$

Proof. The proof comes by making use of (23) and (26) which leads us to $f_k \leq R_k \leq f_{l(k)}$. Then, by employing Lemma 5, the proof is there. □

Lemma 7. Let assume that W_k is not a stationary point of (17). Then, there exists a constant

$$\tilde{\lambda} = \min\left\{1, \frac{2\rho(1-\gamma)}{L\alpha_{\max}}\right\}, \text{ such that } \lambda_k \geq \tilde{\lambda}$$

Proof. In order to prove the lemma, we consider two conditions: the first one is If $\lambda_k \geq 1$, which completes the proof. So, let $\lambda_k < 1$. Now by using the definition of λ_k and (26), we argue that:

$$\begin{aligned} f(W_k + \frac{\lambda_k}{\rho} D_k) \quad &> R_k + \tfrac{\lambda_k}{\rho}\gamma\langle \nabla f(W_k), D_k\rangle \geq \\ &f_k + \tfrac{\lambda_k}{\rho}\gamma\langle \nabla f(W_k), D_k\rangle \end{aligned} \tag{28}$$

where the second inequality is due to the fact that $R_k \geq f_k \forall k \in N_k \cup \{0\}$. Since $\nabla f(W)$ is L-Lipschitz continuous, we have

$$\begin{aligned} &f(W_k + \lambda D_k) - f(W_k) = \\ &\lambda\langle \nabla f(W_k), D_k\rangle + \int_0^\lambda \langle \nabla f(W_k + sD_k) - \nabla f(W_k), D_k\rangle ds \\ &\leq \lambda\langle \nabla f(W_k), D_k\rangle + \int_0^\lambda Ls\|D_k\|^2 ds \\ &= \lambda\langle \nabla f(W_k), D_k\rangle + \frac{L}{2}\lambda^2\|D_k\|^2 \end{aligned} \tag{29}$$

Now, by having (29) and (28) in hand, we can conclude that

$$\frac{\lambda_k}{\rho}\gamma\langle \nabla f(W_k), D_k\rangle \leq \frac{\lambda_k}{\rho}\langle \nabla f(W_k), D_k\rangle + \frac{L}{2}\frac{\lambda_k^2}{\rho^2}\|D_k\|^2,$$

it follows that:

$$\lambda_k \geq \min\left\{1, \frac{2\rho(\gamma-1)}{1}\frac{\langle \nabla f(W_k), D_k\rangle}{\|D_k\|^2}\right\}. \tag{30}$$

Due to the fact that $\langle \nabla f(W_k), D_k\rangle \leq -\frac{1}{\alpha_{\max}}\|D_k\|^2$, we get the proof. □

Henceforth, our next goal is to prove that Algorithm 1 has a global convergence. The following theorem summarizes the global convergence.

Theorem 8. Equipped with assumption $(A1)$ and the first part of lemma 1 for the sequence $\{W_k\}$ returned by Algorithm 1, we have:

$$\lim_{k\to\infty} \|D_k\| = 0. \tag{31}$$

Proof. To prove lemma, we need to show that

$$f_{k+1} \leq R_k - \theta\|D_k\|^2, \tag{32}$$

where β is given by $\theta = \min\left\{\frac{\gamma}{\alpha_{\max}}, \frac{2\gamma\rho(1-\gamma)}{L\alpha_{\max}^2}\right\}$. If $\lambda_k \geq 1$, then we have:

$$\begin{aligned} f_{k+1} \leq & R_k + \gamma\lambda_k\langle\nabla f(W_k), D_k\rangle \leq R_k - \frac{\gamma\lambda_k}{\alpha_{\max}}\|D_k\|^2 \\ & \leq R_k - \frac{\gamma}{\alpha_{\max}}\|D_k\|^2, \end{aligned} \tag{33}$$

which proves (32). Now, assuming $\lambda_k < 1$ and knowing $f_{k+1} \leq R_k + \frac{2\gamma\rho(1-\gamma)}{L\alpha_{\max}}\langle\nabla f(W_k), D_k\rangle$ we have

$$f_{k+1} \leq\leq R_k - \frac{2\gamma\rho(1-\gamma)}{L\alpha_{\max}}\frac{\|D_k\|^2}{\alpha_{\max}}, \tag{34}$$

which demonstrate that (32) holds. Since β is a positive real number, (32) implies that $R_k - f_{k+1} \geq \theta\|D_k\|^2 \geq 0$. This fact along with Lemma 6 proves that $\lim_{k\to\infty}\|D_k\| = 0$. This completes proof and shows that Algorithm 1 benefits from global convergence. □

4 Experiment Results

Here our aim is to confirm the theoretical results and investigate the practical behaviour of Algorithm 1, we also compare the performance of PNBB (two methods i.e., N.M.BB2 and N.M.BB3) with monotone Armijo line search, nonmonotone Armijo suggested in [19], non-monotone algorithm suggested by Grippo [20], the BB method monotone algorithm and the BB non-monotone algorithm suggested by Ahookhosh et.al [19] using synthetic and real datasets. In what follows, we evaluate the perfomance of the proposed PNBB in terms of face recognition ability and efficiency of the algorithm (error, CPU time, the number of objective function evaluations, and gradient behavior).

4.1 Implementation Details

There are several methods to calculate the initial point, for example, random, stochastic random Acol, SVD. In order to speed up the convergence rate of NMF, Boutsidis and Gallopoulos in [28],

suggested Non-negative Double Singular Value decomposition (NNDSVD), which is a new method based on two SVD processes, one to approximate the initial data matrix X and the other to approximate the positive sections of the resulting partial SVD factors but we found empirically that it was too computationally heavy when the number of components k was fairly high ($k > 100$). Herein we used random and SVD initialization. The results for random strategy is obtained by averaging of 10 time of performing that algorithm. In addition, we measure the accuracy of the algorithm in face recognition by $\text{Accuracy} := \frac{\text{The num of faces recognized correctly}}{\text{The total num of test data}} \times 100$.

4.2 Stopping Criterion

There are several strategies for stopping condition, for example, the number of iterations, the value of the gradient, CPU Time, and error. In this paper, here apply the following stopping condition for all algorithms:

$$\begin{aligned} &\|[\nabla_H F(W^k, H^k), \nabla_W F(W^k, H^k)]\|_F \\ &\leq \epsilon \|[\nabla_H F(W^1, H^1), \nabla_W F(W^1, H^1)]\|_F, \end{aligned} \tag{35}$$

where ϵ is a tolerance. Moreover, for both problems (5) and (6), the following stopping condition is used.

$$\|\nabla_W F(W^{k+1}, H^k)\|_F \leq \epsilon_W \tag{36}$$

$$\|\nabla_H F(W^{k+1}, H^{k+1})\|_F \leq \epsilon_H, \tag{37}$$

where

$$\tilde{\epsilon} = \max(10^{-3}, \epsilon) \|[\nabla_H F(W^1, H^1), \nabla_W F(W^1, H^1)]\|_F$$

and $\tilde{\epsilon} = \epsilon_H = \epsilon_W$. Notice that, when each of the algorithms that solve (5) without any iterations, the stopping tolerance is reduced by $\epsilon_W = 0.1\epsilon_W$. Moreover, we use the same strategy for solving (6). Besides, we used 1000 as the maximal number of iterations and 150 seconds as the maximal CPU time for solving sub-problem (5) and (6) in all algorithms. Moreover, we use the following abbreviations for short:
The monotone Armijo method is denoted by "M.Ar."; We use "N1.M.Ar." for non-monotone algorithm suggested by Grippo [20]; The monotone algorithm that uses BB method is presented by "N.MBB2"; The non-monotone algorithm suggested by Ahookhosh et.al [19] that use BB step size is denoted by "N2.M.BB2"; For the new algorithm, we consider two cases for choosing step size, that "BB2" and "BB3". We denoted two cases with "N.M.BB2" and "N.M.BB3" respectively.

4.3 Reconstruction Error Results and Discussions

In this experiment we run the algorithms on a synthetic dataset. Also, we provide all codes in the same subroutine. For comparison of iterative algorithms, Dolan and More [29] proposed a measure comparing the considered algorithms with statistical process by demonstration of performance profiles. In this regard it is known that the performance profile disclose all of the major performance

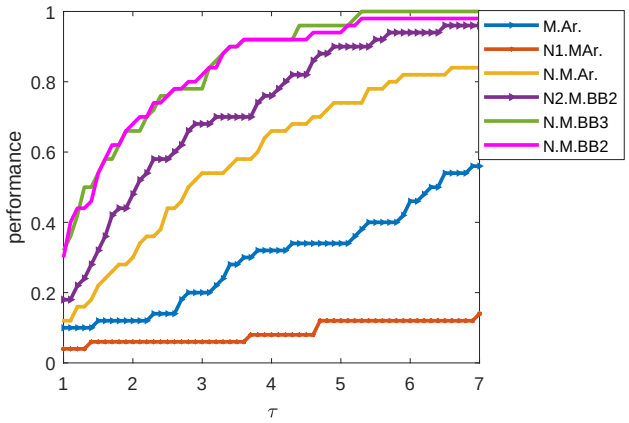

Figure 1: Performance profile for the number of function evaluations.

characteristics, which is a common tool to graphically compare the efficiency of the algorithms. We can use two measures which are, the number of function evaluation and the number of iterations to compare these algorithms. Hence, we use these two indexes for all of the present algorithms separately. Fig. 1 shows the performance of the above algorithms relative to these metrics, respectively. We also can see that PNBB algorithm grows up faster than the other algorithms.

We did the next experiment on the ORL image dataset which contains 400 facial images of 40 different people with 10 images per person. The size of each image is 92×112 pixels, with 256 grey levels per pixel. The matrix X whose columns represent these images has $92 \times 112 = 10,304$ rows and 400 columns. In this experiment, the algorithm use 280 images for training and 120 images for test. We run the algorithm on this dataset and measure the accuracy of the face recognition for various algorithms. The comparison of the accuracy results is presented in Fig. 2 for two different datasets.

In the second set of experiments, we evaluate the performance of the PNBB algorithm in terms of the accuracy of face recognition and the number of times that objective function (the number of inner iterations) is evaluated by the algorithm, CPU time, and the behavior of the gradient. We do the experiment on the ORL dataset and to implement the algorithm, we use the SVD strategy for denoting the starting point and matrix rank. In Fig. 3, we present the CPU time to run PNBB versus different values of tolerances, ϵ on the full dataset. The results reveals that the CPU time of Armijo line search grows fast for low values of the tolerance compared to our PNBB.

The objective function evaluation is another important measure in optimization. Indeed, this term denotes the number of inner iterations. Fewer inner iterations of an algorithm is certainly a true index of its efficiency and showing that it can achieve the solution in less time. Here in Fig. 4, we plotted comparison of different algorithms in terms of the number of objective function evaluation vesus tolerance. As is evident from the figure, the number of inner iteration of the proposed PNBB algorithm is less than that of the Armijo by several orders of magnitude. Further, we plotted the

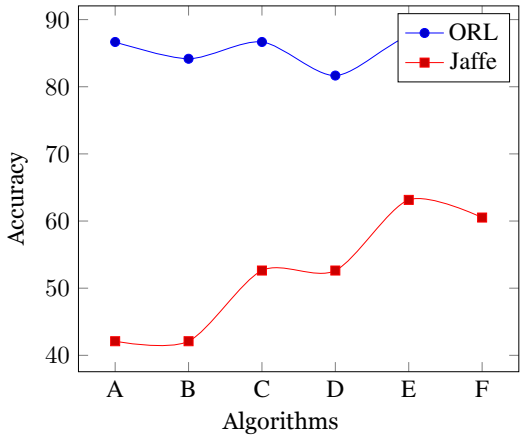

Figure 2: Accuracy results of ORL and Jaffe dataset where both initial point and the matrix rank obtained by SVD approach. Here in this figure, "A", "B", "C", "D", "E" and "F" stand for "M-Ar.", "NM-Ar.", "M-BB2", "NM1-BB2", "NM1-BB3", "NM-BB2", respectively.

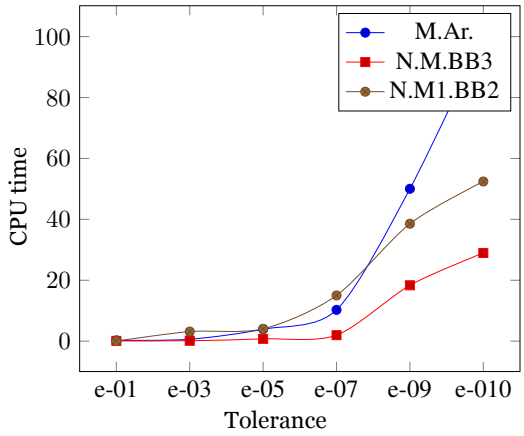

Figure 3: CPU time versus the tolerance (ϵ).

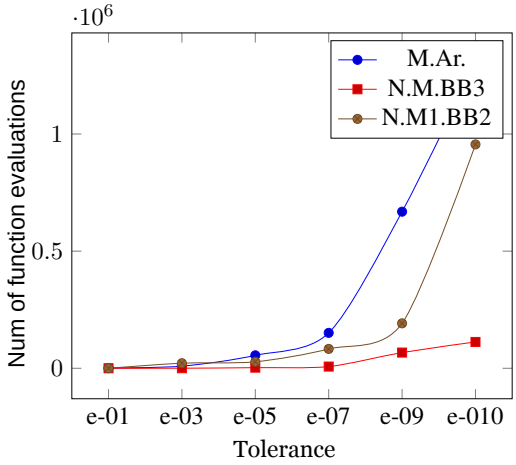

Figure 4: The number of function evaluation versus tolerance (ϵ).

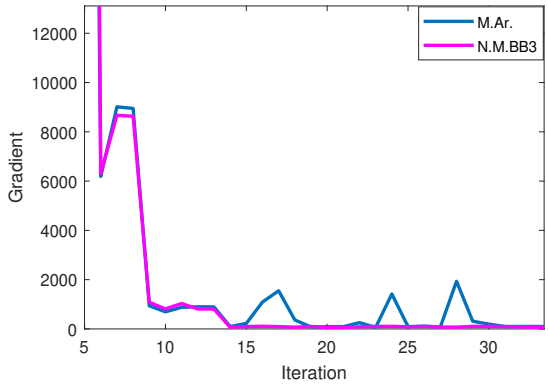

Figure 5: Gradient behaviour of our nonmonotone algorithm compared with that of a monotone one versus the number of iterations.

gradient behaviour of our nonmonotone algorithm compared with that of a monotone one versus the number of iterations in Fig. 5. In our studies the gradient behavior implies the convergence speed of the algorithm. As can be seen from the figure, PNBB as a nonmonotone method converges fast (after around 15 iterations) while the monotone method is still zigzagging after around 30 iterations.

Finally, we now turn our attention to Fig. 6 which shows the error versus the number of iterations. Here error is defined as the difference between the original matrix, X and WH matrix which is indeed the value of the objective function. This figure also reveals that the proposed PNBB outperforms the Armijo algorithm.

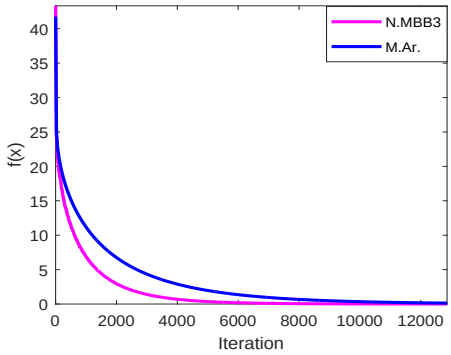

Figure 6: The error (value of the objective function) versus the number of iterations.

5 Conclusion

In this paper, we presented a new efficient nonmonotone BB CG method using the idea of the BB method and some properties of the linear CG method for NMF problems which sequentially optimizes one matrix factor with another fixed. We also develop a generalized Wolfe line search, which is nonmonotone and can avoid a numerical drawback of the original Wolfe line search. Experiments on both synthetic and real-world datasets show that PNBB outperforms existing NMF algorithms in terms of efficiency and overcomes their deficiencies. The face recognition experiments on real-world datasets confirm the effectiveness of PNBB. Since PNBB exploits a suitable nonmonotone method and a proper step size, it accelerates the NMF optimization which is due to the fact that it decreases the number of inner loops for function evaluations.

References

[1] Abhishek Kumar, Vikas Sindhwani, and Prabhanjan Kambadur. Fast conical hull algorithms for near-separable non-negative matrix factorization. In Proceedings of the 30th International Conference on Machine Learning (ICML-13), pages 231–239, 2013.

[2] Daniel D. Lee and H. Sebastian Seung. Algorithms for nonnegative matrix factorization. In Advances in Neural Information Processing Systems (NIPS 13), 2001.

[3] W. Kim, B. Chen, J. Kim, Y. Pan, and H. Park. Sparse nonnegative matrix factorization for protein sequence motif discovery. Expert Systems with Applications, 38(10):13198–13207, 2011.

[4] D. Kuang, H. Park, and C. H. Ding. Symmetric nonnegative matrix factorization for graph clustering. In SDM, volume 12, pages 106–117, 2012.

[5] Han L, Neumann M, Prasad U, Alternating projected Barzilai-Borwein methods for nonneg- ative matrix factorization. Electron Trans Numer Anal 36(6)(2009):5482

[6] Hyunsoo Kim and Haesun Park. Nonnegative matrix factorization based on alternat- ing nonnegativity constrained least squares and active set method. SIAM journal on matrix analysis and applications, 30(2):713–730, 2008.

[7] Andrzej Cichocki, Rafal Zdunek, Anh Huy Phan, and Shun-ichi Amari. Nonnegative matrix and tensor factorizations: applications to exploratory multi-way data analysis and blind source separation. John Wiley Sons, 2009.

[8] Guan N, Tao D, Luo Z, Yuan B, NeNMF: an optimal gradient method for nonnegative matrix factorization. IEEE Trans Signal Process 60(6) (2012):2882–2898

[9] N. D. HO, Nonnegative Matrix Factorization Algorithms and Applications, Ph.D. thesis, FSA/INMA - Departement d'ing enierie math ematique, Universit e Catholique de Louvain, 2008.

[10] N. D. HO, P. VAN DOOREN, AND V.D. BLONDEL, Descent methods for nonnegative matrix factorization, CESAME, Universite Catholique de Louvain, 2008.

[11] A. CICHOCKI, Hierarchical ALS algorithms for nonnegative matrix and 3D tensor factorization, in ICA07, London, UK, September 9-12, Lecture Notes in Comput. Sci., vol. 4666, Springer, Heidelberg, 2007, pp. 169-176.

[12] D. KIM, S. SRA, AND I.S. DHILLON, Fast Newton type methods for the least square nonnegative matrix approximation problem, Proc. SIAM Int'l Conf. Data Mining (SDM'07), eds. S. Parthasarathy and Bing Liu, 2007, pp. 343–354.

[13] D. P. BERTSEKAS, Projected Newton methods for optimization problems with simple constraints, SIAM J. Control Optim., 20 (1982), pp. 221–246.

[14] J. E. Dennis and J. J. More , "Quasi-Newton methods, motivation and theory", Siam Rev. 19 (1977) 46–89; doi:10.1137/1019005.

[15] S. Deng and Z. Wan, "A three-term conjugate gradient algorithm for large-scale unconstrained optimization problems", Appl. Numer. Math. 92 (2015) 70–81; doi:10.1016/j.apnum.2015.01.008.

[16] S. Huang and Z. Wan, "A new nonmonotone spectral residual method for nonsmooth nonlinear equations", J. Comput. Appl. Math. 313 (2017) 82–101; doi:10.1016/j.cam.2016.09.014.

[17] S. Boyd and L. Vandenberghe, Convex Optimization. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[18] R.M. Chamberlain, M.J.D. Powell, C. Lemarechal, H.C. Pedersen, The watchdog technique for forcing convergence in algorithm for constrained optimization Math. Program. Stud., 16 (1982), pp. 1–17

[19] M. Ahookhosh and K. Amini, A nonmonotone trust region method with adaptive radius for unconstrained optimization, Comput. Math. Appl., 60 (2010), 411-422.

[20] Grippo L, Sciandrone M, Nonmonotone globalization techniques for the BarzilaiBorwein gradient method. Comput Optim Appl 23(2)(2002):143–169

[21] K. Amini, M. Ahookhosh, H. Nosratipour, An inexact line search approach using modified nonmonotone strategy for unconstrained optimization, Numer Algor. 66(2014) 49–78

[22] H.C. Zhang, W.W. Hager, A nonmonotone line search technique and its application to unconstrained optimization SIAM J. Optim., 14 (4) (2004), pp. 1043–1056

[23] Huang Y, Liu H, Zhou S (2013) A Barzilai-Borwein type method for stochastic linear complementarity problems. Numer Algor. doi:10.1007/s11075-013-9803-y

[24] Guan N, Tao D, Luo Z, Yuan B (2012) NeNMF: an optimal gradient method for nonnegative matrix factorization. IEEE Trans Signal Process 60(6):2882-2898

[25] J. Barzilai and J.M. Borwein, Two-point step size gradient methods, IMA J. Numer. Anal. 8 (1988), pp. 141-148. doi: 10.1093/imanum/8.1.141

[26] Birgin EG, Martínez JM, Raydan M (2000) Nonmonotone spectral projected gradient methods on convex sets. SIAM J Optim 10(4):1196-1211

[27] A. CICHOCKI, R. ZDUNEK, S. AMARI, Nonnegative matrix and tensor factorization, Signal Processing Magazine, IEEE, 25 (2008), pp. 142–145.

[28] Boutsidis, Christos and Gallopoulos, Efstratios. Svd based ini- tialization: A head start for nonnegative matrix factorization. Pattern Recognition, 41(4):1350–1362, 2008.

[29] Dolan, E., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. 91, 201–213 (2002)