Research Article

A Symmetric Rank-One Quasi-Newton Method for Nonnegative Matrix Factorization

Shu-Zhen Lai,¹ Hou-Biao Li,¹ and Zu-Tao Zhang²

¹ School of Mathematics Science, University of Electronic Science and Technology of China, Chengdu 611731, China

² School of Mechanical Engineering, Southwest Jiaotong University, Chengdu 610031, China

Correspondence should be addressed to Hou-Biao Li; lihoubiao0189@163.com

Received 10 October 2013; Accepted 5 November 2013; Published 22 January 2014

Academic Editors: I. K. Argyros and S. Zhang

Copyright © 2014 Shu-Zhen Lai et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

As is well known, the nonnegative matrix factorization (NMF) is a dimension reduction method that has been widely used in image processing, text compressing, signal processing, and so forth. In this paper, an algorithm on nonnegative matrix approximation is proposed. This method is mainly based on a relaxed active set and the quasi-Newton type algorithm, by using the symmetric rank-one and negative curvature direction technologies to approximate the Hessian matrix. The method improves some recent results. In addition, some numerical experiments are presented in the synthetic data, imaging processing, and text clustering. By comparing with the other six nonnegative matrix approximation methods, this method is more robust in almost all cases.

1. Introduction

An NMF problem is to decompose a nonnegative matrix $V \in \mathbb{R}^{n \times m}$ into two nonnegative matrix $W \in \mathbb{R}^{n \times k}$ and $H \in \mathbb{R}^{k \times m}$, such that WH approximate to V as well as possible. To measure the distance between WH and V , there are many methods such as the Kullback Leibler divergence, and Bregman divergence, and Frobenius divergence. Due to the favorable property of the Frobenius divergence, many methods are presented based on it. The Frobenius divergence may be described as follows:

$$\min_{(W \in \mathbb{R}^{(n \times k)}, H \in \mathbb{R}^{(k \times m)})} f(W, H) = \frac{1}{2} \|V - WH\|_F^2, \quad (1)$$

subject to $W_{ij} \geq 0, H_{ij} \geq 0$ for all i and j .

In the last decade, numerous methods have been proposed to deal with the NMF problem in (1). Most of these methods can be classified into two classes, that is, alternating one-step gradient descent and alternating least squares.

- (i) The alternating one-step gradient is to alternate W and H with one step. The most well known is Lee's multiplicative update algorithm [1], which alternates

W and H by the following rules: suppose we have obtained the l th matrix W^l and H^l , then

$$W_{ia}^{l+1} = W_{ia}^l \frac{(V(H^l)^T)_{ia}}{(W^l H^l (H^l)^T)_{ia}}, \quad \forall i, a, \quad (2)$$

$$H_{bj}^{l+1} = H_{bj}^l \frac{((W^{l+1})^T V)_{bj}}{((W^{l+1})^T W^{l+1} H^k)_{bj}}, \quad \forall b, j.$$

- (ii) For the frame work of the alternating least squares, the procedure is as follows [2]:

- (1) initialize $H \in \mathbb{R}^{k \times m}$ with a nonnegative matrix;
- (2) solve the following problems repeatedly until a stopping criterion is satisfied:

$$\min_{W \geq 0} g(W) = \frac{1}{2} \|H^T W^T - V^T\|_F^2, \quad (3)$$

where H is fixed, and

$$\min_{H \geq 0} g(H) = \frac{1}{2} \|V - WH\|_F^2, \quad (4)$$

where W is fixed.

Obviously, the above two methods both satisfy that

$$\begin{aligned} f(W^{l+1}, H^l) &\leq f(W^l, H^l), \\ f(W^{l+1}, H^{l+1}) &\leq f(W^{l+1}, H^l). \end{aligned} \quad (5)$$

Equation (5) insures that the above numerical methods converge to the solution of (1).

Since (3) and (4) may be regarded as the subproblems of the NMF problem (1). Next we mainly focus on how to solve (4), while (3) is similar to (4), due to the symmetric property of Euclidean distance. There exist some methods to solve (4), such as the projected Newton method [3], quasi-Newton method [4–6], the active set method [7–9], and so forth.

Note that the Frobenius norm of a matrix is just the sum of the Euclidean distance over columns (or rows). Then solving (4) in fact is equal to solving a series of the following nonnegative least squares (NNLS) problems:

$$\begin{aligned} \min_{h_1 \geq 0} g(h_1) &= \frac{1}{2} \|Wh_1 - v_1\|_2^2, \\ \min_{h_2 \geq 0} g(h_2) &= \frac{1}{2} \|Wh_2 - v_2\|_2^2, \dots, \\ \min_{h_k \geq 0} g(h_k) &= \frac{1}{2} \|Wh_k - v_k\|_2^2, \end{aligned} \quad (6)$$

where $h_i, v_i, i = 1, 2, \dots, m$ are the columns of H and V , respectively. But, for convenience, we write the least squares by omitting their subscripts. For example,

$$\min g(\mathbf{h}) = \frac{1}{2} \|W\mathbf{h} - \mathbf{v}\|_2^2, \quad (7)$$

subject to $\mathbf{h} \geq 0$ for all i .

As is well known, if $\mathbf{x}$ is the optimal solution of (7), then $\mathbf{x}$ satisfies the following equation:

$$\begin{aligned} \mathbf{r} &= W^T W \mathbf{h} - W^T \mathbf{v}, \\ \mathbf{r} &\geq 0, \\ \mathbf{x} &\geq 0, \\ \mathbf{x}^T \mathbf{v} &= 0, \end{aligned} \quad (8)$$

which is called the KKT optimal condition [10]. This tells us that one can try to find a descend method whose solutions will satisfy the condition (8) after finite iterations.

The remained part of this paper is organized as follows. In the second section we will propose our algorithm for a single right hand vector nonnegative least squares. In addition, the symmetric rank-one quasi-Newton method for NMF will be discussed in the third section. In the fourth part we present the numerical experiments on both synthetic data and real world data. The last part is to be the conclusion part, in this section we will forecast the future work on NMF.

2. A New Method for Nonnegative Least Squares Problems

The nonnegative least squares can be regarded as a quadratic programming with box constraints, whose upper constraints can be considered as infinite. Recently, Lin proposed the projected algorithm for nonnegative linear square (NNLS) in [3]. Later in [5], Kim et al. proposed the fast projection-based methods (fnmae) for nonnegative least square programming based on the quasi-Newton and active set method. A few years later, Gong and Zhang [11] improved the fnmae algorithm by using the symmetric property of the Hessian matrix of the object function $f(W, H)$. In this paper, we will use the symmetric rank-one quasi-Newton line search technology approximate the Hessian matrix. In addition, we will combine the block active method with the quasi-Newton method and then use the symmetric rank-one technology to modify the BFGS in Kim et al. [5].

2.1. The Frame of Solving NNLS. For the NNLS problem (7), our method is iterative and in each iteration we partition the variables into two parts, namely, the inactive set and the active set. Suppose the current iteration is l , then we are going to compute the $(l + 1)$ th iteration.

Based on the idea of fixed set in [5] and the theory in [12], we denote an inactive set as follows:

$$I_+^l = \{i \mid 0 \leq \mathbf{h}_i^l \leq \epsilon^l, [\nabla g(\mathbf{h})]_i > 0\}, \quad (9)$$

where $\epsilon^l = \min\{\epsilon, \|\mathbf{h}^l - \nabla g(\mathbf{h}^l)\|_F^2\}$, ϵ is a small positive scalar, and $[\nabla g(\mathbf{h})]_i$ denotes the i th element of the gradient $\nabla g(\mathbf{h})$ of the object function $g(\mathbf{h})$.

For convenience, in some places, we will slightly abuse notations and say that $\mathbf{h}_i^l \in I_+^l$ whenever $i \in I_+^l$.

Denote the active variables and the inactive variables at the current iteration by ξ^l and η^l , respectively. Assuming that the $\mathbf{h}^l$ and $\nabla g(\mathbf{h}^l)$ are partitioned as follows:

$$\mathbf{h}^l = \begin{bmatrix} \xi^l \\ \eta^l \end{bmatrix}, \quad (10)$$

where $\xi^l \notin I_+^l$ and $\eta^l \in I_+^l$. That is to say ξ^l is the active part of the current iteration. Then we compute the projection of ξ by the following equation:

$$\xi = P \left[\xi^l - \alpha \bar{D}^l \nabla g(\xi^l) \right], \quad (11)$$

where $\alpha \geq 0$ is the iteration step-size, $\bar{D}^l$ is a gradient scaling matrix, and $P[a]$ is the positive projection, that is,

$$P[a] = \begin{cases} a, & a > 0 \\ 0, & a \leq 0. \end{cases} \quad (12)$$

Next we update the current result $\mathbf{h}^l$ by the following rules:

$$\mathbf{h}^{l+1} = \begin{bmatrix} \xi \\ \eta^l \end{bmatrix} = \begin{bmatrix} P \left[\xi^l - \alpha \bar{D}^l \nabla g(\xi^l) \right] \\ 0 \end{bmatrix}, \quad (13)$$

```

Step 1. start data
    Choose  $\mathbf{h}^0 \geq 0$ , and the maximum iterative number maxiter,
    Set  $l = 0$ ;
Step 2.
    Compute the inactive set by using the rule (9);
Step 3. (update the current solution)
    Renew current solution by (13), where using (16) to approximate
    the inverse of Hessian matrix and (17) to compute the stepsize;
Step 4.
    if the current result reaches the stop criterion then stop, the current
    solution is the final solution;
    else go to Step 2 and Step 3.

```

ALGORITHM 1: The SR1 algorithm for NNLS.

```

Initialization
    Input matrix  $V$ , the maximum number of iteration: maxiter;
    initial matrix  $W^0$ ;
    the approximate rank  $k$ ,  $1 \leq k \leq \min\{m, n\}$ .
for  $i = 1$ : maxiter
    (1)  $W = W^i$ ,  $H^{\text{old}} = H^i$ ;
    (2) Compute the matrix  $H^{\text{new}}$  by using the Algorithm 1, where the object function is (18);
    (3)  $H^{\text{old}} = H^{\text{new}}$ ,  $H^{i+1} = H^{\text{old}}$ ;
    (4)  $H = H^{i+1}$ ,  $W^{\text{old}} = W^i$ ;
        Compute the matrix  $W^{\text{new}}$  by using the Algorithm 1, where the object function is (21);
    (5)  $W^{\text{old}} = W^{\text{new}}$ ,  $W^{i+1} = W^{\text{old}}$ ;
    (6) if the stopping criteria is met, then break.
end for

```

ALGORITHM 2: The SR1 algorithm for NMF.

where the right equation uses the fact that η^l is fixed to zeros, it can be comprehended as that this part satisfies the KKT optimal condition (8), then in the next iteration we need not to update this part. After updating the $\mathbf{h}^{l+1}$, we can compute the $\nabla g(\mathbf{h}^{l+1})$ and update the inactive set I_+^{l+1} to obtain the next iterative result. In the whole iteration we will adhere to the iteration rule until the result satisfies the stop criterions.

2.2. The Approximate to $\bar{D}^l$. As the size of ξ^l and η^l changes at each iteration, the computation of the matrix $\bar{D}^l$ is not an easy task. Note that the curvature information of ξ^l is received from the curvature information of h^l . Due to this fact, the gradient matrix $\bar{D}^l$ can be obtained by taking the proper submatrix of the matrix $\bar{D}$ to make this task easier, where $\bar{D}$ is the matrix that carries the curvature information of the vector $\mathbf{h}^l$. In realization of the method we can try to eliminate the curvature information of η^l . Then the update rule of (13) can be regarded as follows:

$$\mathbf{h}^{l+1} = P \left[\mathbf{h}^l - \alpha \bar{D} \nabla g(\mathbf{h}) \right]. \quad (14)$$

In [5], Kim et al. used the BFGS update method to approximate the Hessian matrix of the object function $g(\mathbf{h})$. The

BFGS method is well established and only uses the gradient information of the object function. But BFGS needs the time consumption. Many researchers have experimentally observed that the symmetric rank-one (SR1) rule performs better than BFGS quasi-Newton update rules [13]. In this paper, we will use the SR1 rule to approximate the Hessian matrix of $g(\mathbf{h})$ to improve the converge speed.

Suppose that H^l is the current approximation of the Hessian matrix, then we apply the following SR1 rule to approximate the next Hessian matrix:

$$H^{l+1} = H^l + \frac{(\mathbf{w}^l - H^l \mathbf{v}^l)(\mathbf{w} - H^l \mathbf{v}^l)^T}{(\mathbf{w} - H^l \mathbf{v}^l)^T \mathbf{w}^l}, \quad (15)$$

where $\mathbf{w}^l = \nabla g(\mathbf{h}^{l+1}) - \nabla g(\mathbf{h}^l)$ and $\mathbf{v}^l = \mathbf{h}^{l+1} - \mathbf{h}^l$. Let D^l be the inverse of the H^l . By using the Sherman-Morrison-Woodbury formula [14], we can obtain the inverse matrix D^{l+1} of H^{l+1} as follows:

$$D^{l+1} = D^l + \frac{(\mathbf{v}^l - D^l \mathbf{w}^l)(\mathbf{v}^l - D^l \mathbf{w}^l)^T}{(\mathbf{v}^l - D^l \mathbf{w}^l)^T \mathbf{w}^l}. \quad (16)$$

FIGURE 1: Relative errors of approximation against iteration number for SR1, fnmae, pnm, alsq, and nnmalq with the matrix $V \in \mathbb{R}^{200 \times 40}$ for the approximate rank $k = 10$.

Thus, we complete the approximate to $\bar{D}^l$ in the iterative processes.

2.3. Line Search Strategy for Step-Size. For (13), $\alpha > 0$, we next consider how to find an α with the largest function reduction. For this purpose, we may choose the searching rule as follows:

$$\alpha = \arg \min_{\alpha \geq 0} g(P[h^l - \alpha \nabla g(h^l)]). \quad (17)$$

To sum up Sections 2.1–2.3, we obtain the following symmetric rank-one NNLS algorithm (see Algorithm 1).

3. A Symmetric Rank-One Quasi-Newton Method for NMF

In the former part, we have discussed the algorithm for NNLS and then in this part we are going to talk about the algorithm for NMF. Note that, in Section 1, we have analyzed that the NMF problem can be resolved into several nonnegative least squares. Due to this property, we can apply the symmetric rank-one method to our NMF problem.

3.1. Applying the SR1 Directly to NNLS Subproblems. Firstly, for (4), we can do vectorization of the matrix H as follows:

$$\begin{aligned} g(H) &= \frac{1}{2} \|V - WH\|_F^2 \\ &= \frac{1}{2} \text{vec}(H)^T \begin{pmatrix} W^T W & & \\ & \ddots & \\ & & W^T W \end{pmatrix} \text{vec}(H) \\ &\quad - \text{tr}((W^T V)^T H) + \frac{1}{2} \text{tr}(V^T V), \end{aligned} \quad (18)$$

FIGURE 2: Relative errors of approximation against iteration number for SR1, fnmae, pnm, AS, alsq, and nnmalq with the matrix $V \in \mathbb{R}^{2000 \times 800}$ for the approximate rank $k = 5$.

where $\text{tr}(B)$ is the trace of matrix B . By using the quasi-Newton idea, the iteration of $\text{vec}(H)$ can be rewritten as

$$\text{vec}(H^{l+1}) = P[\text{vec}(H^l) - \alpha^l \bar{D} \nabla g(H)], \quad (19)$$

where α^l is the step-size of the current iteration, $\bar{D}$ is the gradient scaling matrix, and $\nabla g(H)$ is the gradient of (18).

Equation (18) is actually a nonnegative least square problem; then we can apply the method discussed in the

FIGURE 3: Relative errors of approximation against iteration number for SR1, fnmae, pnm, AS, alsq, and nnmalq with the matrix $V \in \mathbb{R}^{2000 \times 800}$ for the approximate rank $k = 10$.

FIGURE 4: Relative errors of approximation against iteration number for SR1, fnmae, pnm, AS, alsq, and nnmalq with the matrix $V \in \mathbb{R}^{2000 \times 800}$ for the approximate rank $k = 20$.

former part to this problem directly. First let the inactive set of (18) be defined as

$$I_H^l = \{i \mid 0 \leq (\text{vec}(H))_i \leq \epsilon^l, \nabla g(H)_i > 0\}, \quad (20)$$

where ϵ^l can be obtained by the same method as the former part (see Section 2.1). In (19), the gradient scaling matrix $\overline{D}^l$ is approximated by the symmetric rank-one quasi-Newton method. The step-size α^l is done by the line search.

FIGURE 5: Relative errors of approximation against time for SR1 and fnmae with the matrix $V \in \mathbb{R}^{200 \times 40}$ in (a) and $V \in \mathbb{R}^{2000 \times 800}$ in (b) for the approximate rank $k = 10$.

FIGURE 6: Relative errors of approximation against time for SR1, fnmae, alsq, and nnmalq with the matrix $V \in \mathbb{R}^{2000 \times 800}$ for the approximate rank $k = 10$.

FIGURE 7: Reconstructed images after 10 iterations.

FIGURE 8: Reconstructed images after 20 iterations.

For (3), the object function corresponding to W may be also rewritten as

$$\begin{aligned}
 g(W) &= \frac{1}{2} \|V^T - H^T W^T\|_F^2 \\
 &= \frac{1}{2} \text{vec}(H)^T \begin{pmatrix} HH^T & & \\ & \ddots & \\ & & HH^T \end{pmatrix} \text{vec}(W^T) \quad (21) \\
 &\quad - \text{tr}((HV^T)^T W^T) + \frac{1}{2} \text{tr}(V^T V).
 \end{aligned}$$

Thus, we can solve it by the same method as (18).

3.2. The Algorithm for NMF. Next, based on Section 3.1 and the Algorithm 1, we give the symmetric rank-one quasi-Newton algorithm (SR1) for NMF as in Algorithm 2.

Obviously, the computation of subproblems (3) and (4) is high cost. Each subproblem requires an iterative procedure, which can be regarded as a subiteration, and we need to compute the following gradients:

$$\begin{aligned}
 \nabla f_H(W, H) &= W^T W H - W^T V, \\
 \nabla f_W(W, H) &= W H H^T - V H^T.
 \end{aligned} \quad (22)$$

In order to reduce the above computation, we can compute $W^T W$, $W^T V$ and $H^T H$, $V H^T$ in the outer iteration in our method.

FIGURE 9: Reconstructed images after 50 iterations.

3.3. Computational Complexity of Algorithms. According to numerical experiments in Section 4, we find that our algorithm has a slight advantage compared with the faster algorithms such as the fast projection-based method (fnmae) in [5] and alternating nonnegative least squares using the projected Newton method (pnm) [11]. To demonstrate this problem more effectively, next, let us give a simple comparison on their computational complexity.

- (i) For the algorithm SR1, it is very clear that it costs $O(nmk)$ to compute the matrices $W^T W$ and $W^T V$ in the subproblem of NMF and it costs $O(k^2 m^2)$ to compute the gradient scaling matrix and gradient matrix, while the computation of step-size needs $O(sm k^2)$, where s is the iteration number of the step-size. So the computational complexity of the whole method is as follows:

$$O(nmk) + \text{iter} * (O(sm k^2) + O(k^2 m^2)), \quad (23)$$

where the iter is the iteration number, that is, maxiter.

- (ii) The computational complexity of pnm in [11] is

$$O(nmp) + \text{iter} * (O(sm k^2) + O(\tilde{m} \tilde{s})), \quad (24)$$

where s is the iteration number of the step-size and iter is the number of inner iteration. In this method the matrix $\bar{D}$ is expressed by block diagonal matrix, and $\tilde{m}$ is the number of the subblocks and the $\tilde{s}$ is the average column number of the subblocks.

- (iii) The computational complexity of fnmae in [5] is as follows:

$$O(nmk) + \text{iter} * (O(sm k^2) + O(m^3 k^3)), \quad (25)$$

where iter is the iteration number to solve the matrix H and s is the iteration number of solving the step-size.

4. Numerical Experiments

In this part, we present some experiment results from both rand data and real world data and compare our method to

FIGURE 10: Relative error of approximation against the number of iteration for the text data classic 300.

the following six algorithms:

- (1) fnmae: fast projection-based method in [5];
- (2) pnm: alternating nonnegative least squares using the projected Newton method in [11];
- (3) AS: the active set method in [9];
- (4) alsq: the alternating nonnegative least squares method in [2];
- (5) nnmalq: the alternating nonnegative least squares method in [15];
- (6) nmf: the projected gradient method in [3];
- (7) SR1: the symmetric rank-one quasi-Newton method in this paper.

In our experiments, we initialize all the methods randomly and show plots of the relative error against the number of iteration or the time of iteration, where the relative error of approximation is $\|V - WH\|_F / \|V\|_F$.

4.1. Synthetic Data Experiments. By generating the synthetic data randomly, we test the random generated matrix $V \in \mathbb{R}^{n \times m}$ ($n = 200$, $m = 40$) with the approximate rank $k = 10$ and another matrix $V \in \mathbb{R}^{n \times m}$ ($n = 2000$, $m = 800$) with the approximate rank $k = 5, 10$, and 20 , respectively. In our experiments, we yield ten of these random matrices and then we take the average value of the ten experiment data as the final result.

Figure 1 shows the relative error of approximation against iteration number of random matrix $V \in \mathbb{R}^{200 \times 40}$ for the rank $k = 10$ for different methods. As a whole the relative errors of approximation of the SR1, fnmae, and pnm are very similar after a few iterations, but after 80 iterations we can see a slight advantage of our method.

Figures 2, 3, and 4 are the numerical results of the random matrix $V \in \mathbb{R}^{2000 \times 800}$ with the approximate rank $k = 5, 10$, and 20 , respectively. The figures in (a) show the comparisons among the SR1, fnmae, pnm, As, alsq, and nnmalq methods. To have a clear look at similar numerical results of SR1, pnm, and As methods, we present the comparisons of the three methods in (b). From these figures, we can learn that the relative error of approximation of SR1, fnmae, pnm, and As are also similar after a few iterations under the same condition. But our method SR1 shows more robustness than the other ones in all cases.

Figures 5 and 6 are the plots of relative error against the running time. One can see that our method converges much faster than the nnmalq and alsq methods. Comparing with fnmae, our method converges as fast as fnmae at the beginning; after seconds our method shows a slight advantage in the left side of Figure 5.

The above numerical results show that our symmetric rank-one quasi-Newton method improves the efficiency of the quasi-Newton method every iteration and cuts down the time of each iteration. Comparing with alsq and nnmalq, our SR1 method decreases faster in each iteration, while our method has a similar convergence with fnmae, As, and pnm. The nnmalq method decreases slowly in every iteration, so becomes less competitive. In addition, we find that the error of nmf method is much larger than the other above six methods mentioned in our experiment, so we did not plot its error curve in Figures 1–6.

4.2. Application to Imaging Processing. NMF was originally motivated by Lee and Seung [1] in an image processing application. Many others have also considered NMF to image processing, face recognition application, model recognition

FIGURE 11: Relative error of approximation against the number of iteration for the text data classic 3891.

application, signal processing application, and so forth. In this part, we are going to do some numerical experiments on image processing. Our experiment is done on four random chosen faces (<http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html>) and the size of each face image is 92×112 with 256 gray per pixel. We give out the reconstructed image of running 10, 20, and 50 iterations, respectively. In this experiment, we set the approximate rank $k = 14$, and the images are taken randomly from five random initial matrices in each iteration. The first image in each row is the original image; others are the reconstructed images by SR1, fnmae, pnm, AS, alsq, nnmalq, and nmf methods, respectively.

From Figures 7 and 8, we can observe that, after 10 and 20 iterations, SR1 can reconstruct almost the same as fnmae, AS, and pnm procedures and much better than alsq, nnmalq, and nmf procedures. Figure 9 illustrates that the images obtained via SR1, As, fnmae and pnm methods have

similar quality, and the images obtained by alsq are still vague. These pictures show that our method SR1 is also applied to the imaging processing and has a better reconstructing ability.

4.3. Application to Text Clustering. In order to test its ability in the other cases, we are going to apply our method to text clustering problem and compare with the other methods. We show the numerical results of the above methods except for the nmf method on four text data sets, which are high dimensional and sparse. The text data is collected from newspapers (<http://www.shi-zhong.com/software/docdata.zip>). In all the numerical experiments, we let the approximate rank k be the number of classes and we initialize the matrix randomly.

We present the comparisons of the six methods mentioned above except for the nmf method in (a) of Figures 10, 11, 12, and 13. (b) is the convergent specifics on the SR1, AS, and pnm methods. From Figures 10–13, we can know

FIGURE 12: Relative error of approximation against the number of iteration for the text data hitech.

FIGURE 13: Relative error of approximation against the number of iteration for the text data tr45.

that, at the beginning several iterations our method is much better than pnm one, and decrease much faster than the other methods with the same random initial matrix. After five iterations, the convergence of these methods are more and more similar.

5. Conclusion

In this paper, we present an algorithm for NMF problem, it is different from the famous fnmae method [5] in the following aspects.

(1) The active set in our method is a relaxed form, while the active set in [5] is a hard form. The fact that an active set of hard form exhibits undesirable discontinuity at the boundary of the constraint set has been shown in Bertsekas [12], which is not good to the convergence rate, so we use the relax form to avoid this problem.

(2) In addition, the fnmae method [5] uses the BFGS method to approximate the Hessian matrix. However, we apply the symmetric rank-one method to approximate the Hessian matrix. The symmetric rank-one method has been experimentally shown to be better than the BFGS method.

What is more, our method is also different from the pnm method in [11]. The paper [11] uses the symmetric property of the Hessian matrix to approximate the inverse of the Hessian matrix by the Cholesky factorization, while in our paper we approximate the inverse of the Hessian matrix by the symmetric rank-one method. Moreover, some numerical experiments show that the object function decreases more per iteration in our method than in the pnm method [11].

To summarize, we propose a symmetric rank-one quasi-Newton method on the NMF. This method maintains the decrease speed per iteration and decrease the computational time. From the experimental results both the synthetic data and real world data, our method performs very well, which shows more robustness than the other ones in almost all cases. In addition, there are many factors which influence the efficiency of the algorithm for NMF, such as the approximate rank, the stop criterion, and sparseness. On these problems, we will continue researching in the future. Whats more we will also research its usage in the pattern recognition in our next research.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

The authors sincerely thank the authors (see [2, 3, 5, 9, 11, 15]) of the above six methods for their experiments codes and the reviewers and editor for their valuable and detailed comments and suggestions on the early version of this paper, which led to a substantial improvement on the presentation and contents of this paper. The paper was supported by the National Natural Science Foundation of China (11026085, 11101071, 1117105, 11271001, 51175443), the Science and Technology Projects of Sichuan under 2013GZX0150 and the Fundamental Research Funds for China Scholarship Council and the project-sponsored by OATF (UESTC).

References

- [1] D. D. Lee and H. S. Seung, "Learning the parts of objects by non-negative matrix factorization," *Nature*, vol. 401, no. 6755, pp. 788–791, 1999.
- [2] M. W. Berry, M. Browne, A. N. Langville, V. P. Pauca, and R. J. Plemmons, "Algorithms and applications for approximate nonnegative matrix factorization," *Computational Statistics and Data Analysis*, vol. 52, no. 1, pp. 155–173, 2007.
- [3] C.-J. Lin, "Projected gradient methods for nonnegative matrix factorization," *Neural Computation*, vol. 19, no. 10, pp. 2756–2779, 2007.
- [4] R. Zdunek and A. Cichoki, "Non-negative matrix factorization with quasi-Newton optimization," in *Proceedings of the International Conference on Artificial Intelligence and soft Computing (ICAISC '06)*, pp. 870–879, 2006.
- [5] D. Kim, S. Sra, and I. S. Dhillon, "Fast newton-type methods for the least squares nonnegative matrix approximation problem," in *Proceedings of the 7th SIAM International Conference on Data Mining*, pp. 343–354, April 2007.
- [6] D. Kim, S. Sra, and I. S. Dhillon, "Fast projection-based methods for the least squares nonnegative matrix approximation problem," *Statistical Analysis and Data Mining*, vol. 1, no. 1, pp. 38–51, 2008.
- [7] H. Kim and H. Park, "Nonnegative matrix factorization based on alternating nonnegativity constrained least squares and active set method," *SIAM Journal on Matrix Analysis and Applications*, vol. 30, no. 2, pp. 713–730, 2008.
- [8] J. Kim and H. Park, "Toward faster nonnegative matrix factorization: a new algorithm and comparisons," in *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM '08)*, pp. 353–362, December 2008.
- [9] J. Kim and H. Park, "Fast nonnegative matrix factorization: an active-set-like method and comparisons," *SIAM Journal on Scientific Computing*, vol. 33, no. 6, pp. 3261–3281, 2011.
- [10] D. Bertsekas, *Nonlinear Programming*, Athena Scientific, Belmont, Mass, USA, 1999.
- [11] P. Gong and C. Zhang, "Efficient nonnegative matrix factorization via projected Newton method," *Pattern Recognition*, vol. 45, no. 9, pp. 3557–3565, 2012.
- [12] D. Bertsekas, "Projected Newton methods for optimization problems with simple constraints," *SIAM Journal on Control and Optimization*, vol. 20, no. 2, pp. 221–246, 1982.
- [13] F. Öztoprak and Ş. I. Birbil, "A symmetric rank-one quasi-Newton line-search method using negative curvature directions," *Optimization Methods and Software*, vol. 26, no. 3, pp. 455–486, 2011.
- [14] G. H. Golub and C. F. van Loan, *Matrix Computations*, The John Hopkins University Press, 1996.
- [15] D. D. Lee and H. S. Seung, "Algorithms for nonnegative matrix factorization," in *Proceedings of the Neural Information Processing Systems (NIPS '00)*, pp. 556–562, 2000.