International Journal of Signal Processing, Image Processing and Pattern Recognition
Vol.8, No.7 (2015), pp.141-150
http://dx.doi.org/10.14257/ijsip.2015.8.7.13

An Accelerated Iterative Hard Thresholding Method for Matrix Completion

Juan Geng[1], Xingang Yang[2,3], Xiuyu Wang[4] and Laisheng Wang[3*]

[1] College of Mathematics and Statistics, Hebei University of Economics and Business
050064 Shijiazhuang, China
[2] Department of Safety Engineering, China Institute of Industrial Relations
100048 Beijing, China
[3]College of Science, China Agricultural University
100083 Beijing, China
[4]Department of Information Technology, Shijiazhuang Information Engineering Vocational College
050035 Shijiazhuang, China
hebeigengjuan@163.com, xingang2005@126.com, wangxy0311@126.com, wanglaish@126.com

Abstract

The matrix completion problem is to reconstruct an unknown matrix with low-rank or approximately low-rank constraints from its partially known samples. Most methods to solve the rank minimization problem are relaxing it to the nuclear norm regularized least squares problem. Recently, there have been some simple and fast algorithms based on hard thresholding operator. In this paper, we propose an accelerated iterative hard thresholding method for matrix completion (AIHT). Then we report numerical results for solving noiseless and noisy matrix completion problems and image reconstruction. The numerical results suggest that significant improvement can be achieved by our algorithm compared to the other reported methods, especially in terms of CPU time.

Keywords: *Matrix completion, nuclear norm, iterative hard thresholding method, image reconstruction*

1. Introduction

The rank minimization problem states that a low rank unknown matrix $X \in \mathbb{R}^{n_1 \times n_2}$ is to be recovered from a linear measurement $\mathrm{b} = \mathcal{A}(X)$. It has led to many impact applications, such as, image recovery [1, 2], collaborative prediction [3], pattern recognitions [4], etc. One of its special cases is the matrix completion problem, where $\mathcal{A}(\mathrm{X})$ is a subset of the entries of X. In [5, 6], the authors showed that under suitable conditions with very high probability, most $n_1 \times n_2$ matrices of rank r can be perfectly recovered by solving the convex optimization program:

$$\begin{aligned} &\min \ \|X\|_* \\ &\text{s.t. } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \end{aligned} \tag{1}$$

where $M \in \mathbb{R}^{n_1 \times n_2}$ is a matrix with n_1 rows and n_2 columns, $\|X\|_*$ is the nuclear norm of the matrix X, i.e. the sum of its all singular values, Ω is the set of indices of samples and $\mathcal{P}_\Omega$ is the orthogonal projector onto the span of matrices vanishing outside of Ω.

[*] Corresponding author.

Indeed, the nuclear norm minimization (1) can be viewed as a standard linear SDP problem which can be directly solved by interior-point methods and some SDP solvers such as SeDuMi [7] and SDPT3 [8]. Since these methods use second-order information, the memory requirement for computing descent directions quickly becomes too large as the problem size increases. Therefore, some researchers developed a number of efficient methods using only first–order information. Cai *et al.* [9] proposed the singular value thresholding (SVT) algorithm, which is essentially a gradient method for solving the dual of a regularized approximation of (1). In addition, Toh *et al.* [10] studied the accelerated proximal gradient Lagrangian (APGL) method which solves the Lagrangian version of (1), and the proximal point algorithm (PPA) in [11] solving the general nuclear norm minimization problem with linear equality and second-order cone constraints. Recently, Meka *et al.* [12] proposed a simple and fast algorithm SVP (Singular Value Projection) based on the classical projected gradient algorithm, and this method which directly approaches the non-convex rank minimization problem is typically built upon iterative hard thresholding. On the basis of the work of Meka, Tanner *et al.* [13] introduced a simple and efficient alternating projection algorithm NIHT (Normalized Iterative Hard Thresholding) which uses an adaptive stepsize calculated to be exact for a restricted subspace.

The iterative hard thresholding algorithm is actually a gradient descent method and its descent direction is negative gradient direction. The main disadvantage of this method is its slow convergence speed. The motivation of this paper is speed up the iterative hard thresholding algorithm by combining the current gradient and directions of several previous iterative steps and use this new direction as search direction. From the computational results on synthetic data and image in-painting, we can see that our proposed algorithm can obtain more accuracy solution in less time than SVP and NIHT.

The remainder of this paper is organized as follows. Section 2 shows the necessary notations and preliminaries of matrix completion problem. In section 3 the accelerated iterative hard thresholding method for matrix completion (AIHT) is presented. In section 4 we experimentally compare AIHT to SVP and NIHT algorithms on randomly generated matrices and low-rank image recovery problem. Finally, we give a brief summary in Section 5.

2. Notations and Preliminaries on Matrix Completion

Firstly, we briefly introduce some notations used in this paper. As is used before, $X \in \mathbb{R}^{n_1 \times n_2}$ is a $n_1 \times n_2$ matrix of rank r and X_{ij} its *(i, j)*-th entry. u is a vector and u_i its *i*-th component. The inner product of two vectors u and v is denoted by $\langle u, v \rangle$, and the inner product between two matrices is denoted by $\langle X, Y \rangle = \text{trace}(X^* Y)$. The singular values of X are ordered as $\sigma_1(X) \geq \cdots \geq \sigma_i(X) \geq \sigma_{i+1}(X) \geq \cdots \geq \sigma_r(X) > 0$. $\|X\|_*$ denotes the nuclear norm of matrix X and $\|X\|_F$ denotes the Frobenius norm defined by $\|X\|_F^2 = \langle X, X \rangle$. Other notation will be introduced as it occurs.

In this paper, we consider the following problem:

$$\begin{aligned} \min \quad & \varphi(X) = \tfrac{1}{2}\|\mathcal{A}(X) - b\|_2^2 \\ \text{s.t.} \quad & X \in \mathbb{C}(r) = \{X : rank(X) \leq r\}. \end{aligned} \tag{2}$$

Meka *et al.* [12] computed the Euclidean projection onto the non-convex set $\mathbb{C}(r)$ using singular value decomposition, and proposed a simple and fast algorithm SVP (Singular Value Projection) based on the classical projected gradient algorithm. In their method, the classical projected gradient descent update iteration is

$$X^{k+1} = \mathcal{P}_r(X^k - \eta_k \mathcal{A}^*(\mathcal{A}(X^k) - b)), \tag{3}$$

where the $\mathcal{A}^*$ is the adjoint of the linear operator $\mathcal{A}$, and $\mathcal{P}_r(X)$ is called hard thresholding operator and defined as:

$$\mathcal{P}_r(X) := U\Sigma_r V^* \text{ where } \Sigma_r(i,i) := \begin{cases} \Sigma(i,i) & i \le r \\ 0 & i > r. \end{cases} \tag{4}$$

In (3), the stepsize η_k is selected as a fixed constant. Tanner et al. in [13] presented the Normalized IHT (NIHT) algorithm by selecting the stepsize as

$$\mu_k^u = \frac{\|P_u^k \mathcal{A}^*(b-\mathcal{A}(X^k))\|_F^2}{\left\|\mathcal{A}(P_u^k \mathcal{A}^*\left(b-\mathcal{A}(X^k)\right))\right\|_F^2}, \tag{5}$$

where $P_U^k := U_k U_k^*$ and U_k is the top r left singular vectors of X^k. Through careful observations of the update iterations of SVP and NIHT, we discover that the search directions of them are negative gradient direction $\nabla\varphi(X) = \mathcal{A}^*(\mathcal{A}(X) - b)$. However, the main drawback of this method is the need for a large amount of iterations, leading to its slow rate of convergence. In this paper, we apply the so-called linear semi-iterative methods (or polynomial acceleration methods) to the iterative hard thresholding algorithm. The new algorithm combines the current gradient and directions of several previous iterative steps and uses this new direction as search direction. Hence, it accelerates the original iterative hard thresholding algorithm.

3. The Accelerated Iterative Hard Thresholding Method

In 1951, Landweber [14] suggested to solve the following integral equations iteratively,

$$x^k = x^{k-1} + \omega K^*(g - Kx^{k-1}), \quad k = 1, 2, \cdots. \tag{6}$$

where K^* is the adjoint operator of K, x_0 is a initial guess, and $0 < \omega < 2\|K^*K\|^{-1}$. Later, Hanke [15] considered the linear semi-iterative methods for (6). In the semi-iterative method full use is made of all information available at time step k by adding the respective residual to a weighted arithmetic mean of all approximations,

$$x^k = \mu_{1,k}x^{k-1} + \cdots + \mu_{k,k}x^0 + \omega_k K^*(g - Kx^{k-1}), \quad k = 1, 2, \cdots,$$
$$\mu_{1,k} + \cdots + \mu_{k,k} = 1, \quad \omega_k \ne 0. \tag{7}$$

If $\omega_k = \omega$ and $\mu_{2,k} = \cdots = \mu_{k,k} = 0$, (7) is actually the Landweber's method. A special example of semi-iterative is called ν -method [15], the iteration step of which is

$$x^k = x^{k-1} + \mu_k(x^{k-1} - x^{k-2}) + \omega_k K^*(g - Kx^{k-1}) \tag{8}$$

where

$$\mu_1 = 1, \quad \omega_1 = \frac{4\nu+2}{4\nu+1}$$
$$\omega_k = \frac{4(2k+2\nu-1)(k+\nu-1)}{(k+2\nu-1)(2k+4\nu-1)}$$
$$\mu_k = 1 + \frac{(k-1)(2k-3)(2k+2\nu-1)}{(k+2\nu-1)(2k+4\nu-1)(2k+2\nu-3)}$$
$$x^{-1} = x^0. \tag{9}$$

In this paper, we apply the ν -method to the iterative hard thresholding and obtain the update iteration,

$$X^{k+1} = \mathcal{P}_r(X^k + \mu_k\left(X^k - X^{k-1}\right) + \omega_k \mathcal{A}^*(b - \mathcal{A}(X^k))), \tag{10}$$

where $X^{-1} = X^0$ and ω_k, μ_k are shown in (9). In this iteration, the new search direction is the linear combination of the search direction of previous two steps and the current gradient. Thus we obtain an accelerated iterative hard thresholding method for matrix completion (AIHT). Now, based on the above discussion, we proposed the complete AIHT method in Algorithm 1.

Algorithm 1. Accelerated iterative hard thresholding method for matrix completion (AIHT)

```
Input: $M$, $\mathcal{A}$, $b = \mathcal{A}(M)$, $\nu$, $r$, $L$
Initialization: $X^{-1} = X^0 = \mathcal{P}_r(\mathcal{A}^*(b))$
for $k = 1$ to $L$
    1) Compute: $\mu_1 = 1$, $\omega_1 = \frac{4\nu+2}{4\nu+1}$
                $\omega_k = \frac{4(2k+2\nu-1)(k+\nu-1)}{(k+2\nu-1)(2k+4\nu-1)}$, $k \geq 2$
                $\mu_k = 1 + \frac{(k-1)(2k-3)(2k+2\nu-1)}{(k+2\nu-1)(2k+4\nu-1)(2k+2\nu-3)}$, $k \geq 2$
    2) Compute: $\bar{X}^{k+1} = X^k + \mu_k(X^k - X^{k-1}) + \omega_k \mathcal{A}^*(b - \mathcal{A}(X^k))$
    3) Compute: $X^{k+1} = \mathcal{P}_r(\bar{X}^{k+1})$
    4) Repeat steps 2) -3) until stooping criterion is valid.
End for
Output: $X^{opt} = X^{k+1}$.
```

4. Numerical Experiments

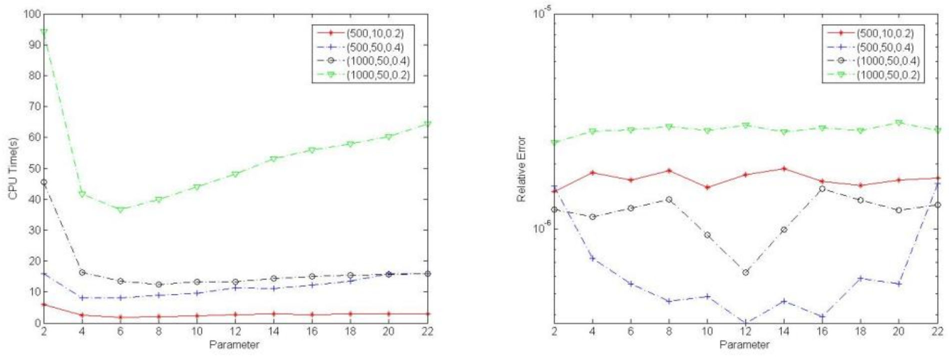
This section is divided into four parts. In the first subsection, we firstly identify a proper value of parameter ν. In the second subsection, we create random matrices and samples sets, and then use our algorithm to solve some examples of the noiseless matrix completion problem. In the third subsection, we present some numerical results on noisy matrix completion problem. In the last subsection, as an application to image processing, we evaluate the effectiveness of our algorithms in low-rank image recovery. We compare AIHT algorithm with the original iterative hard thresholding method (SVP) and the Normalized IHT (NIHT). All results demonstrate our algorithm performs well and more effective. All experiments are performed under MATLAB (Version R2012b), and all the computational results are obtained on a desktop computer with a 3.20GHz CPU and 4 GB of memory.

In our trails, we generate random matrices $M_L \in \mathbb{R}^{n_1 \times r}$ and $M_R \in \mathbb{R}^{n_2 \times r}$, with i.i.d. Guassian entries. Then we set $M = M_L M_R^T$ and sampled randomly a subset Ω of m entries uniformly. In our tests, we used $n_1 = n_2 = n$. We use d_r to denote the "degree of freedom" defined by $r(2n-r)$ for a matrix with rank r, and $SR = m/n^2$ to denote the sampling ratio which means the number of measurements divided by the number of entries of the random matrix. Let X^* be the optimal solution produced by our algorithm, we use the relative error to measure the quality of X^* to original M, i.e.

$$\text{RelErr} = \|X^* - M\|_F / \|M\|_F .$$

4.1 Parameter ν

There is a main parameter ν in AIHT algorithm, hence we have to firstly identify a proper value of ν. For different value of ν, we test our algorithm on four scenarios of (n, r, SR) and to see which one is more suitable. Figure 1 compares the relative error and CPU time on cases $(500,10,0.2)$, $(500,50,0.4)$, $(1000,50,0.2)$ and $(1000,50,0.4)$. From Figure 1 (a), we can see that the four examples use the relatively short CPU time when $8 \leq \nu \leq 10$. And from Figure 1 (b), we also can see that when $10 \leq \nu \leq 12$, there are the best results on the relative errors. This is not a thorough approach to evaluate different ν values, however we find there is an overall nice result when ν around 10. Therefore, considering both solution quality and speed, $\nu = 10$ appeared to be a good choice for our algorithm.

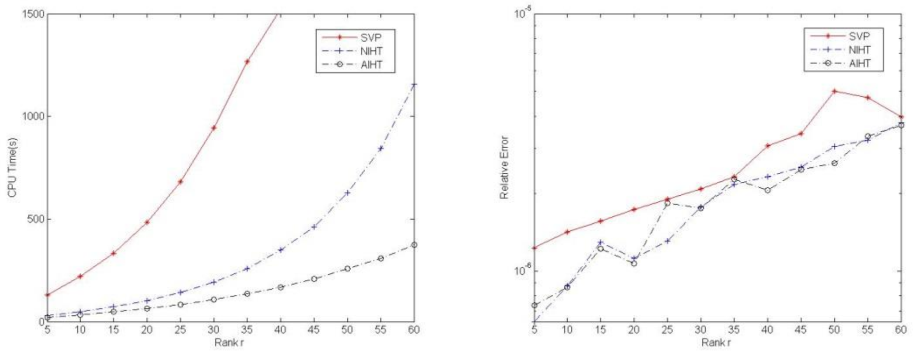


(a) CPU time (s) (b)Relative error

Figure 1. Recovery Results of Four Examples with ν between 2 and 22 (x-axis) by AIHT. (a): CPU Time in Seconds; (b): Relative Error; All Results are Averages of 10 Independent Trials

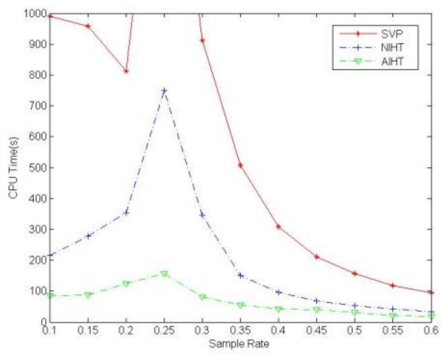
4.2 Numerical Results for Noiseless Matrix Completion

In this subsection, we compare the AIHT algorithm with the original IHT method (SVP) and the Normalized IHT (NIHT). Firstly, we report the results by the three algorithms of using different ranks and percentage of entries. Then, we divide the problems into "easy problems" and "hard problems". By the term "easy problems", we mean problems in which the ratio m/d_r is larger than 3, otherwise problems are called "hard problems". We demonstrate that AIHT algorithm outperforms SVP and NIHT on both easy problems and hard problems.

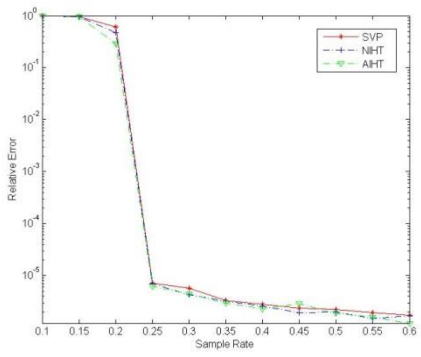


(a) CPU time (s) (b) Relative error

Figure 2. Recovery Results of 1000×1000 Matrices with r between 5 and 60 (x-axis) by AIHT, SVP and NIHT. We choose uniformly at random 20% of the entries of the matrix. (a): CPU time in seconds; (b): relative error. All results are averages of 10 independent trials.



(a) CPU time (s)



(b) Relative error

Figure 3. Recovery results of 1000×1000 matrices with $r = 50$ and SR from 0.1 to 0.6 by AIHT, SVP and NIHT. (a): CPU time in seconds; (b): relative error. All results are averages of 10 independent trials.

In the following two experiments, we compare three algorithms with the sampling ration fixed or the rank fixed. We set $n = 1000$ and a fixed sampling ratio 0.2 in the first test (Figure 2), and a fixed rank 50 in the second test (Figure 3). Figure 2 plots how the three algorithms perform as the rank increases, and Figure 3 plots how they perform as the sampling ratio increases. It is to be expected that, when the sampling ratio is fixed, the complexity of the problem increases with the increase of the rank. On the other hand, when the rank is fixed, the complexity increases with the decrease of the sampling ratio. In terms of the completion speed, AIHT outperforms SVP and NIHT, which can be seen from Figure 2 (a) and Figure 3 (a). In Figure 2 (a), the CPU time used by SVP is more than 1000 seconds when $r > 35$, and NIHT is more than 500 seconds when $r = 60$, while our AIHT algorithm can always obtain the satisfied results in 100 seconds. Compared with the CPU time, the advantage in accuracy of AIHT is not too obvious. In Figure 2 (b), we can observe that NIHT and AIHT perform better than SVP in terms of accuracy, and between AIHT and NIHT, NIHT is slightly inferior. In the second experiment, the accuracy of the three methods is strikingly similar.

Table 1. Numerical Results of Easy Problems

Objectives	SVP		NIHT		AIHT	
(n, r, m/d_r)	T	RelErr	T	RelErr	T	RelErr
(200, 5, 6)	7	2.12e-6	2	1.62e-6	1	9.36e-7
(200, 10, 5)	5	1.69e-6	2	1.26e-6	2	5.33e-7
(200, 15, 3)	12	2.40e-6	4	2.25e-6	4	5.00e-7
(500, 10, 6)	12	1.71e-6	3	1.32e-6	1.5	1.20e-6
(500, 20, 5)	9	1.65e-6	3	1.47e-6	1.4	6.76e-7
(500, 30, 3)	21	2.12e-6	7	1.97e-6	3	1.36e-6
(800, 10, 6)	262	1.80e-6	46	1.32e-6	33	1.16e-6
(800, 30, 5)	203	1.59e-6	59	1.17e-6	33	1.17e-6
(800, 50, 3)	504	2.05e-6	145	1.88e-6	71	1.22e-6
(1000, 10, 6)	101	1.79e-6	20	1.68e-6	11	1.40e-6
(1000, 50, 5)	50	1.56e-6	27	1.41e-6	19	6.43e-7
(1000, 60, 3)	148	2.02e-6	45	1.87e-6	20	1.75e-6

Table 2. Numerical Results of Hard Problems

Objectives	SVP		NIHT		AIHT	
(n, r, m/d_r)	T	RelErr	T	RelErr	T	RelErr

(200, 5, 2.5)	125	6.43e-6	21	6.38e-6	6	4.05e-6
(200, 10, 2)	68	4.44e-6	15	4.39e-6	5	4.00e-6
(200, 15, 1.7)	86	5.07e-6	22	4.99e-6	7	4.95e-6
(500, 10, 2.5)	95	3.63e-6	15	3.69e-6	6	3.35e-6
(500, 30, 2)	74	3.37e-6	18	3.12e-6	7	2.63e-6
(500, 50, 1.7)	104	3.83e-6	33	3.56e-6	9	2.83e-6
(800, 10, 2.5)	1609	4.20e-6	265	4.06e-6	144	4.08e-6
(800, 30, 2)	1861	3.98e-6	379	3.06e-6	160	3.63e-6
(800, 50, 1.7)	2045	4.62e-6	658	3.92e-6	251	3.90e-6
(1000, 10, 2.5)	866	3.85e-6	89	3.85e-6	38	3.41e-6
(1000, 50, 2)	460	3.13e-6	109	2.89e-6	43	2.61e-6
(1000, 60, 1.7)	2280	3.91e-6	941	3.71e-6	311	3.50e-6

In next two tests, we decide to test the three algorithms on two special types of problems --"easy problems" and "hard problems", which are mentioned in the beginning of this subsection. The average results of 10 independent trials are listed in Table 1 and Table 2. In the tables, we use "T" to denote the CPU time of every algorithm.

In Table 1 we report the numerical results of easy problems by AIHT, SVP and NIHT for different scenarios of $(n, r, m/d_r)$. As can be seen, in cases (200, 5, 6), (200, 10, 5), (200, 15, 3), (200, 20, 5) and (1000, 50, 5), AIHT is able to complete the test matrices with higher accuracy than others. In the other cases, the accuracy of the three methods is almost the same. In terms of running time, the CPU time of AIHT is far less than the other two methods, and the SVP algorithm takes the most time. Similar phenomenon can be observed in Table 2, which demonstrates the comparison results on hard problems. The accuracy of the three algorithms is similar, and all can be reach to about 10^{-6}. However the amount of time required by the three methods is dramatically different, and AIHT performs better.

In general, the above simulated examples show that AIHT is superior to the SVP and NIHT, since the former has the better reconstruction property and faster speed.

4.3 Numerical Results for Noisy Matrix Completion

In this subsection, we take the noisy case into consideration. As regards matrix completion with noise, Candès et al. [16] gave novel results showing that matrix completion is provably accurate even when the few observed entries are corrupted with a small amount of noise. We tested this problem by using the AIHT algorithm, and compare it with the SVP and NIHT methods. Firstly, we created random matrices and sampled sets as above. Then we corrupted the observations M_{ij} by noise as in the following model:

$$B_{ij} = M_{ij} + \sigma\Sigma_{ij}, \ (i, j) \in \Omega$$

where the noise matrix Σ is an $n \times n$ matrix with i.i.d. Gaussian entries and $\sigma > 0$ is a constant. For our numerical experiments, we take $\sigma = 10^{-2}$ and

$$\varepsilon = \frac{1}{10}\| (\sigma\,\Sigma)_\Omega \|_F .$$

Table 3. Numerical Results for Noisy Data ($\sigma = 10^{-2}$)

Objectives	SVP		NIHT		AIHT	
(n, r, m/d$_r$)	T	RelErr	T	RelErr	T	RelErr
(200, 5, 6)	3	4.50e-3	1.3	4.37e-3	1.2	4.36e-3
(200, 10, 5)	3	4.84e-3	1.7	4.81e-3	1.5	4.81e-3
(200, 15, 3)	5	6.61e-3	2.5	6.53e-3	2	6.53e-3
(500, 10, 6)	29	4.61e-3	13	4.44e-3	10	4.39e-3

(500, 20, 5)	29	4.89e-3	26	4.85e-3	12	4.83e-3
(500, 30, 3)	60	6.86e-3	27	6.73e-3	19	6.64e-3
(800, 10, 6)	114	4.80e-3	37	4.42e-3	31	4.41e-3
(800, 30, 5)	100	4.85e-3	54	4.81e-3	40	4.81e-3
(800, 50, 3)	216	6.76e-3	98	6.64e-3	70	6.63e-3
(1000, 10, 6)	211	5.08e-3	63	4.48e-3	47	4.46e-3
(1000, 50, 5)	185	4.77e-3	122	4.76e-3	99	4.76e-3
(1000, 60, 3)	403	6.77e-3	178	6.66e-3	129	6.66e-3

The stopping criterion is as same as the beginning of this section. The computational results from AIHT, SVP and NIHT are shown in Table 3. From there, we see that all the three methods work well, as the relative error is just about equal to σ. By comparison, it is evident that the AIHT performs better than SVP and NIHT as regards CUP time. Similarly to the noiseless matrix completion problem, the accuracy of the three algorithms is similar, and all can be reach to about 10^{-3}.

4.4 Application in Low-rank Image Recovery

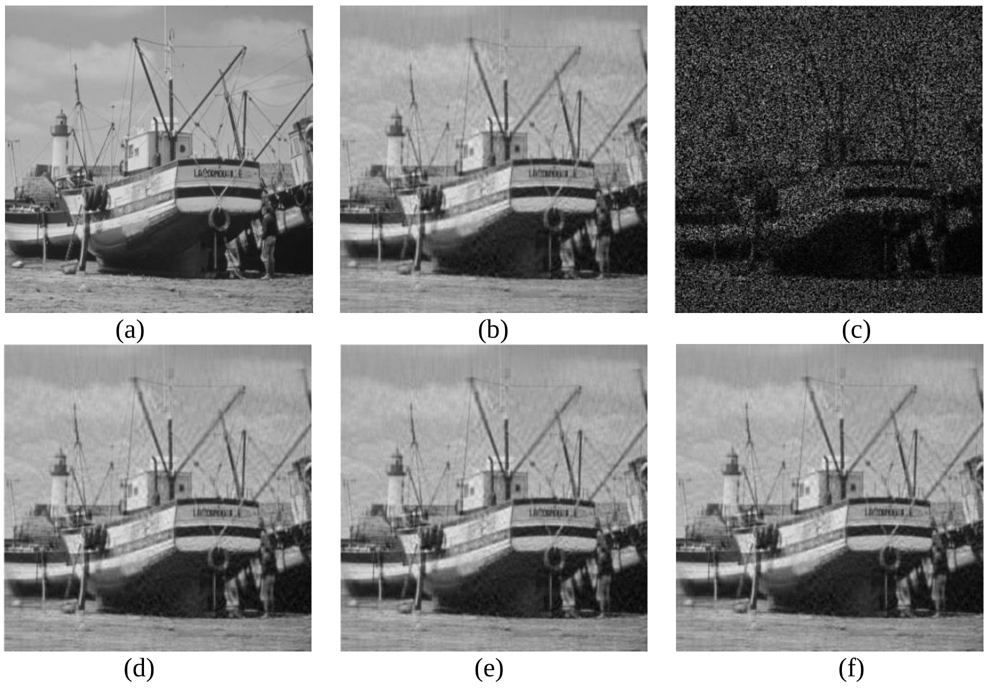


Figure 4. "Boat" image. (a) Original 512×512 image with full rank. (b) Original image truncated to be rank 40. (c) Selected randomly 30% samples from image (b). (d) Recovered image from (c) by SVP algorithm. (e) Recovered image from (c) by NIHT algorithm. (f) Recovered image from image (c) by AIHT algorithm.

As an application to image processing, we evaluate the effectiveness of SVP, NIHT and AIHT algorithms in low-rank image recovery. In this experiment, we test the image "Boat" that has 512×512 pixels with full rank and has been widely used in many simulations, since the image has a nice mixture of details, shading area, flat regions, and textures. In Figure 4, we used SVD to the original image (a) and truncated this decomposition to get the low rank-40 image (b). Comparing with the original image (a), the low-rank image (b) loses some details. We randomly selected 30% samples from image (b) and obtain image (c). Then, we recover it by the three algorithms respectively.

The recovered image is image (d)-(f). In this test, $m/d_r = 1.99$, hence it is a hard problem, while the three algorithms all provide good recovery effect, and the relative error is about 10^{-6}. Table 4 lists the numerical results of the three methods for recovering the grayscale image. We can see from the table that although the similar precision by the three methods, but the CPU time is different remarkably. AIHT is 6 times faster than the NIHT and 13 times than SVP to achieve the same accuracy. From these tests, it is obvious that our AIHT algorithm can effectively recover the details of the low-rank image (b) in less time.

Table 4. Numerical Results of the Image "Boat"

	T	RelErr
SVP	937	7.68e-6
NIHT	441	8.66e-6
AIHT	72	3.02e-6

5. Conclusion

The existing algorithms based on hard thresholding operator in matrix completion are mostly gradient descent methods and their descent directions are all negative gradient directions. However, the main drawback of these methods is the need for a large amount of iterations, leading to its slow rate of convergence. In this paper, we apply the so-called linear semi-iterative methods (or polynomial acceleration methods) to the original iterative hard thresholding algorithm and obtain the new algorithm namely accelerated iterative hard thresholding method (AIHT), which combines the current gradient and directions of several previous iterative steps. In numerical comparisons, we experimentally compare the proposed method with SVP and NIHT algorithms. Thorough simulations, we show that the AIHT algorithm is faster than others in the same reconstruction performance. In our future work, we will investigate the case of large-scale by some other techniques and test its performance for image reconstruction problems.

Acknowledgements

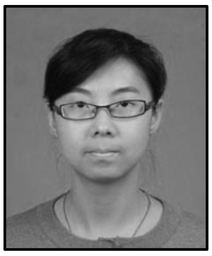
This work was supported by the National Natural Science Foundation of China (No. 11271367).

References

[1] [1] T. Morita and T. Kanade, "A sequential factorization method for recovering shape and motion from image streams", Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 19, no. 8, **(1997)**, pp. 858-867.
[2] [2] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method", International Journal of Computer Vision, vol. 9, no. 2, **(1992)**, pp. 137-154.
[3] [3] J. D. M. Rennie and N. Srebro, "Fast maximum margin matrix factorization for collaborative prediction", Proceedings of the 22nd international conference on Machine, **(2005)**, pp. 713-719.
[4] [4] L. Eldén, "Matrix Methods in Data Mining and Pattern Recognition (Fundamentals of Algorithms)", SIAM, Philadelphia, PA, USA, **(2007)**
[5] [5] B. Recht, M. Fazel and P.A. Parrilo, "Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization", SIAM Review, vol. 52, **(2010)**, pp. 471-501.
[6] [6] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization", Foundations of Computational Mathematics, vol. 9, no. 6, **(2009)**, pp. 717-772.
[7] [7] J. F. Sturm, "Using SeDuMi 1.02, a Matlab toolbox for optimization over symmetric cones", Optimization Methods and Software, vol. 11, no. 1, **(1999)**, pp. 625–653.
[8] [8] R. H. TüTüNCü, K. C. Toh and M. J. Todd, "Solving semidefinite-quadrtic-linear programs using SDPT3", Mathematical Programming, vol. 95, **(2003)**, pp. 189–217.
[9] [9] J. F. Cai, E. J. Candès and Z. W. Shen, "A singular value thresholding algorithm for matrix completion", SIAM Journal on Optimization, vol. 20, no. 4, **(2010)**, pp. 1956–1982.

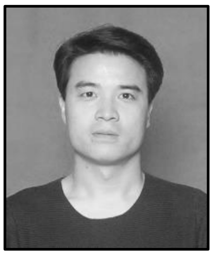
[10] [10] K. C. Toh and S. W. Yun, "An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems", Pacific Journal of Optimization, vol. 6, **(2010)**, pp. 615-640.
[11] [11] Y. J. Liu, D. F. Sun and K. C. Toh, "An implementable proximal point algorithmic framework for nuclear norm minimization", Mathematical Programming, vol. 133, **(2012)**, pp. 399-436.
[12] [12] R. Meka, P. Jain and I. S. Dhillon, "Guaranteed rank minimization via singular value projection", Advances in Neural Information Processing Systems, **(2010)**, pp. 937–945.
[13] [13] J. Tanner and K. Wei, "Normalized iterative hard thresholding for matrix completion", Proceedings available online at http://people.maths.ox.ac.uk/tanner/papers/TaWei_NIHT.pdf , **(2013)**
[14] [14] L. Landweber, "An iteration formula for Fredholm integral equations of the first kind", American journal of mathematics, **(1951)**, pp. 615-624.
[15] [15] M. Hanke, "Accelerated Landweber iterations for the solution of ill-posed equations", Numerische mathematik, vol. 60, no. 1, **(1991)**, pp. 341-373.
[16] [16] E. J. Candès, Y. Plan, "Matrix completion with noise", Proc IEEE, vol. 98, **(2010)**, pp. 925-936.

Authors



Juan Geng, She received her B. S. degree from Hebei Normal University, China (2003) and M. S. degree from Hebei Normal University, China (2006). She received the Ph.D. degree from China Agriculture University, Beijing, China, in 2014. Her research interests include matrix rank minimization, tensor completion and image processing.

Xingang Yang, He received the B. S. degree from Agricultural University of Hebei (2005), China and M. S. degree from Capital Normal University, China (2008). Currently, he is a Ph.D. student in China Agriculture University, China. His research interests include matrix rank minimization, meachine learning and image processing.



Xiuyu Wang, He received the B. S. degree from Hebei Normal University, China (2003) and M. S. degree from Hebei Normal University, China (2009). Currently, he is a instructor at Shijiazhuang Information Engineering Vocational College, Hebei, China. His research interests include data mining and machine learning.

Laisheng Wang, He received the B. S. degree from Jilin University, China (1982) and M. S. degree from Jilin University, China (1989). He received the Ph.D. degree from China Agricultural University, Beijing, China, in 2001. Currently, he is a professor at the school of science, China Agricultural University, Beijing, China. His research interests include data mining, image processing and pattern recognition.