Anefficientthresholding Algorithm for Matrix Completion

E. Fathima and K.P. Kaliyamoorthie

Department of Computer Science, Bharath University, Chennai-73, India

Abstract: This paper introduces a novel algorithm to approximate the matrix with minimum nuclear norm among all matrices obeying a set of convex constraints. This problem may be understood as the convex relaxation of a rank minimization problem and arises in many important applications as in the task of recovering a large matrix from a small subset of its entries (the famous Netflix problem). Off-the-shelf algorithms such as interior point methods are not directly amenable to large problems of this kind with over a million unknown entries. This paper develops a simple rst-order and easy-to-implement algorithm that is extremely efficient at addressing problems in which the optimal solution has low rank. The algorithm is iterative and produces a sequence of matrices \( X_k; Y_k \) and at each step, mainly performs a soft-thresholding operation on the singular values of the matrix \( Y_k \). There are two remarkable features making this attractive for low-rank matrix completion problems. The rst is that the soft-thresholding operation is applied to a sparse matrix; the second is that the rank of the iterates \( X_k \) is empirically nondecreasing. Both these facts allow the algorithm to make use of very minimal storage space and keep the computational cost of each iteration low. On the theoretical side, we provide a convergence analysis showing that the sequence of iterates converges. On the practical side, we provide numerical examples in which 1;000 x 1;000 matrices are recovered in less than a minute on a modest desktop computer. We also demonstrate that our approach is amenable to very large scale problems by recovering matrices of rank about 10 with nearly a billion unknowns from just about 0.4% of their sampled entries. Our methods are connected with the recent literature on linearized Bregman iterations for \( \ell_1 \) minimization and we develop a framework in which one can understand these algorithms in terms of well-known Lagrange multiplier algorithms.

Key words: Nuclear norm minimization • Matrix completion • Singular value thresholding • Lagrange dual function • Uzawa's algorithm

INTRODUCTION

Algorithm Outline: Because minimizing the nuclear norm both provably recovers the lowest-rank matrix subject to constraints and gives generally good empirical results in a variety of situations, it is understandable of great interest to develop numerical methods for solving (1.1). In this optimization problem was solved using one of the most advanced semidefinite programming solvers, namely, SDPT3 [1]. This solver and others like SeDuMi are based on interior-point methods and are problematic when the size of the matrix is large because they need to solve huge systems of linear equations to compute the Newton direction. In fact, SDPT3 can only handle \( n \) matrices with \( n \leq 100 \). Presumably, one could resort to iterative solvers such as the method of conjugate gradients to solve for the Newton step but this is problematic as well since it is well known that the condition number of the Newton system increases rapidly as one gets closer to the solution. In addition, none of these general purpose solvers use the fact that the solution may have low rank. We refer the reader to [2] for some recent progress on interior-point methods concerning some special nuclear norm-minimization problems [3].

This paper develops the singular value thresholding algorithm for approximately solving the nuclear norm minimization problem and by extension, problems of the form where \( A \) is a linear operator acting on the space of \( n \) \( n \) matrices and \( b \) 2 \( R \). This algorithm is a simple rst-order method and is especially well suited for problems of very large sizes in which the solution has low rank [4]. We sketch this algorithm in the special matrix completion setting and let \( P \) be the orthogonal projector onto the span of matrices vanishing outside of so that the \( i; j \)th
component of $P(X)$ is equal to $X_{ij}$ if $(i; j) \in 2$ and zero otherwise. Our problem may be expressed as with optimization variable $X \in R^{n_1 \times n_2}$. Fix $\delta > 0$ and a sequence $\delta_{kgk1}$. Then starting with $Y_0 = 0$, $\delta_{2 \times n_1 \times n_2}$, the algorithm inductively $(X_k = \text{shrink}(Y_{k-1}^2); Y_k = Y_{k-1}^2 + \delta P(M X_k)$. In Section 3, we extend the SVT algorithm and formulate a general iteration which is applicable to general convex constraints. In Section 4, we establish the convergence results for the iterations given in Sections 2 and 3. We demonstrate the performance and effectiveness of the algorithm through numerical examples in Section 5 and review additional implementation details. Finally, we conclude the paper with a short discussion in Section 6.

Before continuing, we provide here a brief summary of the notations used throughout the paper. Matrices are bold capital, vectors are bold lowercase and scalars or entries are not bold. For instance, $X$ is a matrix and $X_{ij}$ its $(i; j)$th entry [9]. Likewise, $x$ is a vector and $x_i$ its $i$th component. The nuclear norm of a matrix is denoted by $kXk$, the Frobenius norm by $kXk_F$ and the spectral norm by $kXk_2$; note that these are respectively the 1-norm, the 2-norm and the sup-norm of the vector of singular values. The adjoint of a matrix $X$ is $X^*$ and similarly for vectors. The notation $\text{diag}(x)$, where $x$ is a vector, stands for the diagonal matrix with $x_i$ as diagonal elements. We denote by $hX; Y_i = \text{trace}(X^* Y)^2$ the standard inner product between two matrices ($kXk_2^2 = hX; Xi$). The Cauchy-Schwarz inequality gives $hX; Y_i kXk_2 kYk_2$ and it is well known that we also have $hX; Y_i kXk_2^2 Y_i kXk_2^2$ (the spectral and nuclear norms are dual from one another).

This section introduces the singular value thresholding algorithm and discusses some of its basic properties. We begin with the definition of a key building block, namely, the singular value thresholding operator.

**The Singular Value Shrinkage Operator:** Consider the singular value decomposition (SVD) of a matrix $X \in R^{n_1 \times n_2}$ of rank $r$

$$X = U \Sigma V^* = \text{diag}(f g l i \ r);$$

where $U$ and $V$ are respectively $n_1 \times r$ and $n_2 \times r$ matrices with orthonormal columns and the singular values $i$ are positive (unless specified otherwise, we will always assume that the SVD of a matrix is given in the reduced form above). For each $\delta$, we introduce the soft-thresholding operator $D$ defined as follows: where $t^+$ is the positive part of $t$, namely, $t^+ = \max(0; t)$. In words, this operator simply applies a soft-thresholding rule to the singular values of $X$, effectively shrinking these towards zero. This is the reason why we will also refer to this transformation as the singular value shrinkage operator [10]. Even though the SVD may not be unique, it is easy
to see that the singular value shrinkage operator is well defined and we do not elaborate further on this issue. In some sense, this shrinkage operator is a straightforward extension of the soft-thresholding rule for scalars and vectors. In particular, note that if many of the singular values of X are below the threshold, the rank of D(X) may be considerably lower than that of X, just like the soft-thresholding rule applied to vectors leads to sparser outputs whenever some entries of the input are below threshold.

The singular value thresholding operator is the proximity operator associated with the nuclear norm. Details about the proximity operator can be found in e.g. [5, 7, 8] for some recent applications in the area of image inpainting and image restoration. Just as iterative soft-thresholding methods are designed to find sparse solutions, our iterative singular value thresholding scheme is designed to find a sparse vector of singular values. In classical problems arising in the areas of compressed sensing and signal or image processing, the sparsity is expressed in a known transformed domain and soft-thresholding is applied to transformed coefficients. In contrast, the shrinkage operator D is adaptive. The SVT not only discovers a sparse singular vector but also the bases in which we have a sparse representation. In this sense, the SVT algorithm is an extension of earlier iterative soft-thresholding schemes.

Relation with Other Works: Our algorithm is inspired by recent work in the area of 1 minimization and especially by the work on linearized Bregman iterations for compressed sensing, see [11] for linearized Bregman iterations and [12] for some information about the end of compressed sensing. In this line of work, linearized Bregman iterations are used to find the solution to an underdetermined system of linear equations with minimum 1 norm. In fact, Theorem 2.1 asserts that the singular value thresholding algorithm can be formulated as a linearized Bregman iteration. Bregman iterations were first introduced in as a convenient tool for solving computational problems in the imaging sciences and a later paper [13] showed that they were useful for solving 1-norm-minimization problems in the area of compressed sensing. Linearized Bregman iterations were proposed in to improve performance of plain Bregman iterations, see also [14]. Additional details together with a technique for improving the speed of convergence called kicking are described in. On the practical side, the paper [15] applied Bregman iterations to solve a deblurring problem while on the theoretical side, the references gave a rigorous analysis of the convergence of such iterations. New developments keep on coming out at a rapid pace and recently, introduced a new iteration, the split Bregman iteration, to extend Bregman-type iterations (such as linearized Bregman iterations) to problems involving the minimization of 1-like functionals such as total-variation norms, Besov norms and so forth.

When applied to 1-minimization problems, linearized Bregman iterations are sequences of soft-thresholding rules operating on vectors. Iterative soft-thresholding algorithms in connection with 1 or total-variation minimization have quite a bit of history in signal and image processing and we would like to mention the works for total-variation minimization, for 1-minimization and
shrinkage operator is by-and-large the dominant cost in the SVT algorithm, we expect that a Fortran implementation would run about 3 to 4 times faster.

As for most SVD packages, though one can specify the number of singular values to compute, PROPACK can not automatically compute only those singular values exceeding the threshold. One must instead specify the number s of singular values ahead of time and the software will compute the s largest singular values and corresponding singular vectors. To use this package, we must then determine the number sk of singular values of Y k 1 to be computed at the kth iteration. We use the following simple method. Let rk 1 = rank(Xk 1) be the number of nonzero singular values of Xk 1 at the previous iteration. Set sk = rk 1 +1 and compute the rst sk singular values of Y k 1. If some of the computed singular values are already smaller than, then sk is a right choice. Otherwise, increment sk by a pre-defined integer ‘ repeatedly until some of the singular values fall below. In the experiments, we choose ‘ = 5. Another rule might be to repeatedly multiply sk by a positive number|e.g. 2|until our criterion is met. Incrementing sk by a fixed integer works very well in practice; in our experiments, we very rarely need more than one update.

We note that it is not necessary to rerun the Lanczos iterations for the rst sk vectors since they have been already computed; only a few new singular values (of them) need to be numerically evaluated. This can be done by modifying the PROPACK routines. We have not yet modified PROPACK, however. Had we done so, our run times would be decreased.

**Step Sizes:** There is a large literature on ways of selecting a step size but for simplicity, we shall use step sizes that are independent of the iteration count; that is k = for k = 1; 2;:::. From Theorem 4.2, convergence for the completion problem is guaranteed provided that 0 << 2. This choice is, however, too conservative and the convergence is typically slow. In our experiments, we use instead i.e. 1:2 times the undersampling ratio. We give a heuristic justification below. Secondly, using the equality criterion is met. Incrementing sk by a fixed integer works very well in practice; in our experiments, we very rarely need more than one update.

We note that it is not necessary to rerun the Lanczos iterations for the rst sk vectors since they have been already computed; only a few new singular values (of them) need to be numerically evaluated. This can be done by modifying the PROPACK routines. We have not yet modified PROPACK, however. Had we done so, our run times would be decreased.

<table>
<thead>
<tr>
<th>Table 1: Unknown M</th>
<th>Computational results</th>
</tr>
</thead>
<tbody>
<tr>
<td>size (n n)</td>
<td>rank (r)</td>
</tr>
<tr>
<td>1; 000</td>
<td>1; 000</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>5; 000</td>
<td>5; 000</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>10; 000</td>
<td>10; 000</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>20; 000</td>
<td>20; 000</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>30; 000</td>
<td>30; 000</td>
</tr>
</tbody>
</table>

where M is the real unknown matrix. All of these quantities are averaged over ve runs. The table also gives the percentage of entries that are observed, namely, m=n2 together with a quantity that we may want to think as the information oversampling ratio. Recall that an n n matrix of rank r depends upon dr: = r(2n r) degrees of freedom.
Then \( m = dr \) is the ratio between the number of sampled entries and the 'true dimensionality' of an \( n \times n \) matrix of rank \( r \). The first observation is that the SVT algorithm performs extremely well in these experiments. In all of our experiments, it takes fewer than 200 SVT iterations to quickly reach the value \( r \) of the true rank. After these few initial steps, the SVT iterations search for that matrix with rank \( r \) minimizing the objective functional. As a consequence, the run times are short. As indicated in the table, we note that one recovers a 1; 000 1; 000 matrix of rank 10 in less than a minute. The algorithm also recovers 30; 000 30; 000 matrices of rank 10 from about 0.4% of their sampled entries in just about 17 minutes. In addition, higher-rank matrices are also efficiently completed: for example, it takes between one and two hours to recover 10; 000 10; 000 matrices of rank 100 and 20; 000 20; 000 matrices of rank 50. We would like to stress that these numbers were obtained on a modest CPU (1.86GHz). Furthermore, a Fortran implementation is likely to cut down on these numbers by a multiplicative factor typically between three and four.

We also check the validity of the stopping criterion by inspecting the relative error defined in the table shows that the heuristic and nonrigorous analysis of Section 5.1 holds in practice since the relative reconstruction error is of the same order as \( kP(X_{opt}M)F = kPMF \) 104. Indeed, the overall relative errors reported in Table 1 are all less than 2 10 4.

We emphasized all along an important feature of the SVT algorithm, which is that the matrices \( X_k \) have low rank. We demonstrate this fact empirically in Figure 1, which plots the rank of \( X_k \) versus the iteration count \( k \) and does this for unknown matrices of size 5; 000 5; 000 with different ranks. The plots reveal an interesting phenomenon: in our experiments, the rank of \( X_k \) is nondecreasing so that the maximum rank is reached in the final steps of the algorithm. In fact, the rank of the iterates quickly reaches the value \( r \) of the true rank. After these few initial steps, the SVT iterations search for that matrix with rank \( r \) minimizing the objective functional. As mentioned earlier, the low-rank property is crucial for making the algorithm run fast.

**Linear Inequality Constraints:** We now examine the speed at which one can solve similar problems with linear inequality constraints instead of linear equality constraints. We assume the model, where the matrix \( M \) of rank \( r \) is sampled as before and solve the problem by using. We formulate the inequality constraints in with \( E_{ij} = 0 \) so that one searches for a solution \( M \) with minimum nuclear norm among all those matrices whose sampled entries deviate from the observed ones by at most the noise level. 2 In this experiment, we adjust to be one tenth of a typical absolute entry of \( PM \), i.e. \( = 0.1 ij2 jM_{ij} = m \) and
the noise ratio as defined earlier is 0.780. We set $n = 1,000$, $r = 10$ and the number $m$ of sampled entries is $ve$ times the number of degrees of freedom, i.e. $m = 5dr$. Just as before, we set $= 5n$ and choose a constant step size $= 1.2\pi 1$. The results, reported in Figure 2, show that the algorithm behaves just as well with linear inequality constraints. To make this point, we compare our results with those obtained from noiseless data (same unknown matrix and sampled locations). In the noiseless case, it takes about 150 iterations to reach the tolerance $= 10^{-4}$ whereas in the noisy case, convergence occurs in about 200 iterations (Figure 2(a)). In addition, just as in the noiseless problem, the rank of the iterates is nondecreasing and quickly reaches the true value $r$ of the rank of the unknown matrix 2 This may not be conservative enough from a statistical viewpoint but this works well in this case and our emphasis here is on computational rather than statistical issues.

We close by pointing out that from a statistical point of view, the recovery of the matrix $M$ from undersampled and noisy entries by the matrix equivalent of the Dantzig selector appears to be accurate since the relative error obeys $M M F = M F = 0.0769$ (recall that the noise ratio is about 0.08).

**DISCUSSION**

This paper introduced a novel algorithm, namely, the singular value thresholding algorithm for matrix completion and related nuclear norm minimization problems. This algorithm is easy to implement and surprisingly effective both in terms of computational cost and storage requirements when the minimum nuclear-norm solution is also the lowest-rank solution. We would like to close this paper by discussing a few open problems and research directions related to this work.

Our algorithm exploits the fact that the sequence of iterates $X^k$ have low rank when the minimum nuclear solution has low rank. An interesting question is whether one can prove (or disprove) that in a majority of the cases, this is indeed the case.

It would be interesting to explore other ways of computing $D(Y)$ in words, the action of the singular value shrinkage operator. Our approach uses the Lanczos bidiagonalization algorithm with partial reorthogonalization which takes advantages of sparse inputs but other approaches are possible. We mention two of them.

- A series of papers have proposed the use of randomized procedures for the approximation of a matrix $Y$ with a matrix $Z$ of rank. When this approximation consists of the truncated SVD retaining the part of the expansion corresponding to singular values greater than, this can be used to evaluate $D(Y)$. Some of these algorithms are efficient when the input $Y$ is sparse [41] and it would be interesting to know whether these methods are fast and accurate enough to be used in the SVT iteration.
- A wide range of iterative methods for computing matrix functions of the general form $f(Y)$ are available today, see for a survey. A valuable research direction is to investigate whether some of these iterative methods, or other to be developed, would provide powerful ways for computing $D(Y)$.

In practice, one would like to solve for large values of. However, a larger value of generally means a slower rate of convergence. A good strategy might be to start with a value of, which is large enough so that admits a low-rank solution and at the same time for which the algorithm converges rapidly. One could then use a continuation method as in to increase the value of sequentially according to a schedule $0; 1;:::$ and use the solution to the previous problem with $= i 1$ as an initial guess for the solution to the current problem with $= i$ (warm starting). We hope to report on this in a separate paper.

**ACKNOWLEDGMENTS**

J-F. C. is supported by the Wavelets and Information Processing Programme under a grant from DSTA, Singapore. E. C. is partially supported by the Waterman Award from the National Science Foundation and by an ONR grant N00014-08-1-0749. Z. S. is supported in part by Grant R-146-000-113-112 from the National University of Singapore. E. C. would like to thank Benjamin Recht and Joel Tropp for fruitful conversations related to this project and Stephen Becker for his help in preparing the computational results of Section 5.2.2.

**REFERENCES**