Low rank matrix completion by alternating steepest descent methods

Jared Tanner[a], Ke Wei[b]

[a]*Mathematical Institute, University of Oxford, Andrew Wiles Building, Radcliffe Observatory Quarter, Woodstock Road, Oxford OX2 6GG*
[b]*Department of Mathematics, Hong Kong University of Science and Technology, Hong Kong*

Abstract

Matrix completion involves recovering a matrix from a subset of its entries by utilizing interdependency between the entries, typically through low rank structure. Despite matrix completion requiring the global solution of a non-convex objective, there are many computationally efficient algorithms which are effective for a broad class of matrices. In this paper, we introduce an alternating steepest descent algorithm (ASD) and a scaled variant, ScaledASD, for the fixed-rank matrix completion problem. Empirical evaluation of ASD and ScaledASD on both image inpainting and random problems show they are competitive with other state-of-the-art matrix completion algorithms in terms of recoverable rank and overall computational time. In particular, their low per iteration computational complexity makes ASD and ScaledASD efficient for large size problems, especially when computing the solutions to moderate accuracy such as in the presence of model misfit, noise, and/or as an initialization strategy for higher order methods. A preliminary convergence analysis is also presented.

Keywords: Matrix completion, alternating minimization, gradient descent, exact line-search
2000 MSC: 15A29, 41A29, 65F10, 65J20, 68Q25, 90C26

1. Introduction

The problem of recovering a low rank matrix from partial entries - also known as matrix completion - arises in a wide variety of practical context, such as model reduction [17], pattern recognition [8], and machine learning [1, 2]. From the pioneering work on low rank approximation by Fazel [10] and matrix completion by Candes and Recht [6], this problem has received intensive investigations both from theoretical and algorithmic aspects, see [3, 4, 5, 7, 12, 14, 16, 18, 19, 20, 21, 23, 24, 25] and references therein for a partial review. These matrix completion and low rank approximation techniques rely on the dependencies between entries imposed by the low rank structure. Explicitly seeking the lowest rank matrix consistent with the known entries is expressed as

$$\min_{Z\in\mathbb{R}^{m\times n}} \text{rank}(Z), \text{ subject to } P_\Omega(Z) = P_\Omega(Z^0), \tag{1}$$

where $Z^0 \in \mathbb{R}^{m\times n}$ is the underlying matrix to be reconstructed, Ω is a subset of indices for the known entries, and P_Ω is the associated sampling operator which acquires only the entries indexed by Ω. Problem (1) is non-convex and generally NP-hard [13] due to the rank objective. One of the most widely studied approaches is to replace the rank objective in (1) with its convex relaxation, the Schatten 1-norm (also known as nuclear norm) which is the sum of the singular values, i.e. $\min_{Z\in\mathbb{R}^{m\times n}} \|Z\|_*$, subject to $P_\Omega(Z) = P_\Omega(Z^0)$. It has been proven that, provided the singular vectors are weakly correlated with the canonical basis, the solution of (1) can be obtained by solving the aforementioned convex relaxation [6]. Alternative to the convex relaxation, there have been many algorithms which are designed to attempt to solve for the global minima of (1) directly; many of them are adaptations of algorithms for compressed sensing, such as the hard thresholding algorithms [3, 14, 16, 21]. The iterative thresholding algorithms typically update a current rank r estimate along a line-search direction which departs the manifold of rank r matrices, and then projects back to the rank

Email addresses: `tanner@maths.ox.ac.uk` (Jared Tanner), `makwei@ust.hk` (Ke Wei)

r manifold by computing the nearest matrix in Frobenius norm. The most direct implementation of these algorithms require computing a partial singular value decomposition (SVD) in each iteration. The computational complexity of computing the SVD has complexity of $O(n^3)$ when r, m and n are proportional, causing computing the SVD to be the dominant computational cost per iteration and limits their applicability for large n. A subclass of methods which further exploit the manifold structure in their line-search updates achieve superior efficiency, for $r \ll n$, of order $O(r^3)$. Examples include LRGeomCG [23] which is a nonlinear conjugate gradient method, ScGrassMC [20] which uses a scaled gradient in the subspace update, and the algorithms presented in [18, 19] which use other geometries and metrics.

To circumvent the high computational cost of an SVD, other algorithms explicitly remain on the manifold of rank r matrices by using the factorization $Z = XY$ where $X \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{r\times n}$. Based on this simple factorization model, rather than solving (1), algorithms are designed to solve the non-convex problem $\min_{X,Y} f(X, Y)$ where

$$f(X, Y) := \frac{1}{2}\|P_\Omega(Z^0) - P_\Omega(XY)\|_F^2. \tag{2}$$

Algorithms for the solution of (2) usually follow an alternating minimization scheme, with PowerFactorization [12] and LMaFit [24] two representatives. We present alternating steepest descent (ASD), and a scaled variant ScaledASD, which incorporate an exact line-search to update the solutions for the model (2). *In so doing ASD and ScaledASD have a lower per iteration complexity than PowerFactorization, allowing a more efficient exploration of the manifold in the early iterations where the current estimate is inaccurate. Moreover, ASD and ScaledASD are able to recover matrices of substantially higher rank than can LMaFit.*

The manuscript is outlined as follows. In Sec. 2 we briefly review the alternating minimization algorithms PowerFactorization and LMaFit which motivate ASD. In Sec. 3 we propose ASD for (2), which replaces the least square subproblem solution in PowerFactorization with exact line-search so as to reduce the per iteration computational cost. In Sec. 4 we present ScaledASD, a version of ASD which is accelerated by scaling the search directions adaptively to improve the asymptotic convergence rate. A preliminary convergence analysis for ASD and ScaledASD are given in Sec. 3 and Sec. 4 respectively. Numerical experiments presented in Sec. 5 contrast the aforementioned algorithms and show that ScaledASD is highly efficient, particularly for large problems when solved to moderate accuracy.

2. Alternating minimization

Alternating minimization is widely used for optimization problems due to its simplicity, low memory requirement and flexibility [9]. Without loss of generality, let us consider an objective function $f(X, Y)$ with two variable components $X \in \mathcal{X}, Y \in \mathcal{Y}$, where $\mathcal{X}$ and $\mathcal{Y}$ are the admissible sets. The alternating minimization method, also known as Gauss-Seidel or 2-block coordinate descent method, minimizes $f(X, Y)$ by successive searches over $\mathcal{X}$ and $\mathcal{Y}$. For problem (2), $\mathcal{X} := \mathbb{R}^{m\times r}, \mathcal{Y} := \mathbb{R}^{r\times n}$. PowerFactorization (PF), Alg. 1, is a matrix completion algorithm which applies the alternating minimization method to (2).

Algorithm 1 PowerFactorization (PF, [12])

```
Input: $P_\Omega(Z^0)$, $X_0 \in \mathbb{R}^{m\times r}$, $Y_0 \in \mathbb{R}^{r\times n}$
Repeat
    1. Fix $Y_i$, solve $X_{i+1} = \arg\min_X \|P_\Omega(Z^0) - P_\Omega(XY_i)\|_F^2$
    2. Fix $X_{i+1}$, solve $Y_{i+1} = \arg\min_Y \|P_\Omega(Z^0) - P_\Omega(X_{i+1}Y)\|_F^2$
Until termination criteria is reached
```

Algorithm 2 Low-rank Matrix Fitting (LMaFit, [24])

```
Input: $P_\Omega(Z^0)$, $X_0 \in \mathbb{R}^{m\times r}$, $Y_0 \in \mathbb{R}^{r\times n}$
Repeat
    1. $X_{i+1} = Z_i Y_i^\dagger = \arg\min_X \|XY_i - Z_i\|_F^2$
    2. $Y_{i+1} = X_{i+1}^\dagger Z_i = \arg\min_Y \|X_{i+1}Y - Z_i\|_F^2$
    3. $Z_{i+1} = X_{i+1}Y_{i+1} + P_\Omega(Z^0 - X_{i+1}Y_{i+1})$
Until termination criteria is reached
```

Note that each subproblem in Alg. 1 is a standard least square problem. In [12], a rank increment technique is also proposed to improve the performance of the algorithm. As a typical alternating minimization method, limit points of

Alg. 1 are necessarily stationary points [12]. The per iteration computational cost of PF is determined by the solution of the least squares subproblems in Alg. 2. To decrease this per iteration cost, [24] proposes the model

$$\min_{X,Y,Z} \frac{1}{2}\|Z - XY\|_2^2, \text{ subject to } P_\Omega(Z) = P_\Omega(Z^0), \tag{3}$$

where the projection onto Ω is moved from the objective in (2) to a linear constraint. The alternating minimization approach applied to (3) gives the low-rank matrix fitting algorithm (LMaFit), Alg. 2. LMaFit obtains new approximate solutions of X and Y by solving least square problems while Z is held fixed, as done in PF. However the subproblems in LMaFit have explicit solutions, which reduces the high per iteration computational cost of PF. Moreover, [24] proposes using an over-relaxation scheme to further accelerate Alg. 2. The convergence of limit points to stationary points is similarly established for LMaFit, see Thm. 3.5 in [24].

3. Alternating steepest descent

Solving the least square subproblem in PF to high accuracy is both computationally expensive and potentially of limited value when the factor that is held fixed is inconsistent with the known entries $P_\Omega(Z^0)$. To improve the computational efficiency we replace solving the least squares subproblems in PF with a single step of simple line-search along the gradient descent directions. The alternating steepest descent method (ASD) applies steepest gradient descent to $f(X,Y)$ in (2) alternatively with respect to X and Y. If $f(X,Y)$ is written as $f_Y(X)$ when Y is held constant and $f_X(Y)$ when X is held constant, the directions of gradient ascent are

$$\nabla f_Y(X) = -(P_\Omega(Z^0) - P_\Omega(XY))Y^T \quad \text{and} \quad \nabla f_X(Y) = -X^T(P_\Omega(Z^0) - P_\Omega(XY)). \tag{4}$$

The steepest descent stepsizes along the gradient descent directions can be computed explicitly.

Lemma 3.1. *Let t_x, t_y be the steepest descent stepsizes for descent directions $-\nabla f_Y(X)$ and $-\nabla f_X(Y)$, then*

$$t_x = \frac{\|\nabla f_Y(X)\|_F^2}{\|P_\Omega(\nabla f_Y(X)Y)\|_F^2} \quad \textit{and} \quad t_y = \frac{\|\nabla f_X(Y)\|_F^2}{\|P_\Omega(X\nabla f_X(Y))\|_F^2}. \tag{5}$$

Proof. We only present a proof for t_x and the proof for t_y can be similarly established. First note that

$$t_x = \arg\min_t g(t) \quad \text{where} \quad g(t) = \frac{1}{2}\|P_\Omega(Z^0) - P_\Omega((X - t\nabla f_Y(X))Y)\|_2^2.$$

So $g'(t_x) = 0$. Differentiating $g(t)$ and setting $g'(t_x) = 0$ gives

$$\begin{aligned}
&\langle P_\Omega(Z^0) - P_\Omega((X - t_x\nabla f_Y(X))Y), \nabla f_Y(X)Y\rangle \\
&= \langle P_\Omega(Z^0) - P_\Omega(XY), \nabla f_Y(X)Y\rangle + t_x\langle P_\Omega(\nabla f_Y(X)Y), \nabla f_Y(X)Y\rangle \\
&= \langle (P_\Omega(Z^0) - P_\Omega(XY))Y^T, \nabla f_Y(X)\rangle + t_x\langle P_\Omega(\nabla f_Y(X)Y), P_\Omega(\nabla f_Y(X)Y)\rangle \\
&= -\langle \nabla f_Y(X), \nabla f_Y(X)\rangle + t_x\langle P_\Omega(\nabla f_Y(X)Y), P_\Omega(\nabla f_Y(X)Y)\rangle = 0,
\end{aligned} \tag{6}$$

where the second to last equality follows from the expression for $\nabla f_Y(X)$. The formula for t_x follows from (6). □

3.1. Per iteration computational cost: efficient residual updates

Algorithm 3 consists of two main parts: the computations of the gradient and the steepest descent stepsize. Forming the gradient involves a matrix product between the residual and an $m \times r$ or $r \times n$ matrix. The computations of both the residual and the matrix product require to leading order $2|\Omega|r$ floating point operations (flops), where $|\Omega|$ denotes the number of sampled entries. Computing the stepsize also requires $2|\Omega|r$ flops for the denominator. Naively implemented this would require a per iteration complexity for Alg. 3 of $12|\Omega|r$ flops. However the residual only needs to be computed once at the beginning of an iteration and then can be updated efficiently. Taking the first two steps as an example, the residual after X_i is updated to X_{i+1} is

$$P_\Omega(Z^0) - P_\Omega(X_{i+1}Y_i) = P_\Omega(Z^0) - P_\Omega((X_i - t_{x_i}\nabla f_{Y_i}(X_i))Y_i) = P_\Omega(Z^0) - P_\Omega(X_iY_i) + t_{x_i}P_\Omega(\nabla f_{Y_i}(X_i)Y_i).$$

Fortunately, $P_\Omega(\nabla f_{Y_i}(X_i)Y_i)$ is formed when computing t_{x_i}, so the residual can be updated without computing a matrix-matrix product; therefore the leading order cost per iteration of Alg. 3 is $8|\Omega|r$ flops.

Algorithm 3 Alternating Steepest Descent (ASD)

```
**Input**: $P_\Omega(Z^0),\ X_0 \in \mathbb{R}^{m\times r},\ Y_0 \in \mathbb{R}^{r\times n}$
**Repeat**
    1. $\nabla f_{Y_i}(X_i) = -(P_\Omega(Z^0) - P_\Omega(X_iY_i))Y_i^T$, $t_{x_i} = \frac{\|\nabla f_{Y_i}(X_i)\|_F^2}{\|P_\Omega(\nabla f_{Y_i}(X_i)Y_i)\|_F^2}$
    2. $X_{i+1} = X_i - t_{x_i}\nabla f_{Y_i}(X_i)$
    3. $\nabla f_{X_{i+1}}(Y_i) = -X_{i+1}^T(P_\Omega(Z^0) - P_\Omega(X_{i+1}Y_i))$, $t_{y_i} = \frac{\|\nabla f_{X_{i+1}}(Y_i)\|_F^2}{\|P_\Omega(X_{i+1}\nabla f_{X_{i+1}}(Y_i))\|_F^2}$
    4. $Y_{i+1} = Y_i - t_{y_i}\nabla f_{X_{i+1}}(Y_i)$
**Until** termination criteria is reached
```

3.2. ASD convergence analysis

The convergence analysis of block coordinate methods for general unconstrained optimization problems has been studied in [11, 25] based on stationary points with the boundedness conditions on the search stepsizes. As is typical for global convergence of algorithms for non-convex problems, convergence from an arbitrary starting point to stationary points is shown. Due to the special structure of problem (2), we are able to offer a direct proof that limit points of ASD with the stepsize selection rules (5) are necessarily stationary points of (2); that is limit points (X, Y) satisfy

$$\nabla f_X(Y) = 0 \quad \text{and} \quad \nabla f_Y(X) = 0. \tag{7}$$

Theorem 3.2. *Any limit point of the sequence* (X_i, Y_i) *generated by Alg. 3 is a stationary point.*

The proof of Thm. 3.2 follows that of Thm. 4.1 as Alg. 3 can be viewed as a special case of Alg. 4 with an identity scaling matrix. The fact that limit points of PowerFactorization are necessarily stationary points can be similarly obtained by noting that the least squares update decreases (2) more than the steepest descent update [22].

4. Scaled alternating steepest descent

In this section, we present an accelerated version of Alg. 3. To motivate its construction, let us consider the case when all the entries of Z^0 are observed. Under this assumption, problem (2) can be simplified to

$$\min_{X,Y} \frac{1}{2}\|Z^0 - XY\|_F^2. \tag{8}$$

The Newton directions for problem (8) with respect to X and Y are $(Z^0 - XY)Y^T(YY^T)^{-1}$ and $(X^TX)^{-1}X^T(Z^0 - XY)$, which are the gradient descent directions scaled by $(YY^T)^{-1}$ and $(X^TX)^{-1}$ respectively. However when only partial entries of Z^0 are known, the Newton directions do not possess explicit formulas similar to (4), and solving each subproblem by Newton's method is the same as solving a least square problem as $f(X, Y)$ is an exact quadratic function of X or Y. Scaled ASD, Alg. 4, is a newton like method using the gradient descent directions scaled by $(YY^T)^{-1}$ and $(X^TX)^{-1}$ with exact line-search. The steepest descent stepsizes in ScaledASD can be computed similarly as in ASD, see Lem. 3.1.

Algorithm 4 Scaled Alternating Steepest Descent (ScaledASD)

```
**Input**: $P_\Omega(Z^0),\ X_0 \in \mathbb{R}^{m\times r},\ Y_0 \in \mathbb{R}^{r\times n}$
**Repeat**
    1. $\nabla f_{Y_i}(X_i) = -(P_\Omega(Z^0) - P_\Omega(X_iY_i))Y_i^T$
    2. $d_{x_i} = -\nabla f_{Y_i}(X_i)(Y_iY_i^T)^{-1}$, $t_{x_i} = -\langle \nabla f_{Y_i}(X_i), d_{x_i}\rangle / \|P_\Omega(d_{x_i}Y_i)\|_F^2$
    3. $X_{i+1} = X_i + t_{x_i}d_{x_i}$
    4. $\nabla f_{X_{i+1}}(Y_i) = -X_{i+1}^T(P_\Omega(Z^0) - P_\Omega(X_{i+1}Y_i))$
    5. $d_{y_i} = (X_{i+1}^TX_{i+1})^{-1}\nabla f_{X_{i+1}}(Y_i)$, $t_{y_i} = -\langle \nabla f_{X_{i+1}}(Y_i), d_{y_i}\rangle / \|P_\Omega(X_{i+1}d_{y_i})\|_F^2$
    6. $Y_{i+1} = Y_i + t_{y_i}d_{y_i}$
**Until** termination criteria is reached
```

In the computation of the scaled gradient descent direction d_{x_i}, it requires $2(m+n)r^2+O(r^3)$ flops to compute $Y_iY_i^T$, $(Y_iY_i^T)^{-1}$ and $-\nabla f_{Y_i}(X_i)\cdot(Y_iY_i^T)^{-1}$ one after another. Similarly the computational of of d_{y_i} also requires $2(m+n)r^2+O(r^3)$ flops. Thus the leading order per iteration computational cost of Alg. 4 is $8|\Omega|r + 4(m+n)r^2$ since the residual can be updated as in ASD. For example, after X_i is updated to X_{i+1}, we have

$$P_\Omega(Z^0) - P_\Omega(X_{i+1}Y_i) = P_\Omega(Z^0) - P_\Omega((X_i - t_{x_i}d_{x_i})Y_i) = P_\Omega(Z^0) - P_\Omega(X_iY_i) + t_{x_i}P_\Omega(d_{x_i}Y_i).$$

And again $P_\Omega(d_{x_i}Y_i)$ has already been computed in the calculation of t_{x_i}.

4.1. ScaledASD convergence analysis

To state a similar result to Thm. 3.2 for ScaledASD, we require the limit point is nondegenerate.

Theorem 4.1. *Let (X_i, Y_i) be the sequence generated by Alg. 4. Assume that (X_i, Y_i) are nonsingular (i.e. full rank) during all the iterations. Let (X_{i_k}, Y_{i_k}) be a subsequence of (X_i, Y_i) such that*

$$\lim_{i_k\to\infty} X_{i_k} = X^* \quad \textit{and} \lim_{i_k\to\infty} Y_{i_k} = Y^*$$

for a nonsingular pair of (X^*, Y^*). Then (X^*, Y^*) is a stationary point, that is (X^*, Y^*) satisfy (7).

In order to prove (X^*, Y^*) is a stationary point, by continuity of the gradient, it is sufficient to prove

$$\lim_{i_k\to\infty} \nabla f_{X_{i_k}}(Y_{i_k}) = 0 \quad \text{and} \quad \lim_{i_k\to\infty} \nabla f_{Y_{i_k}}(X_{i_k}) = 0,$$

which follows immediately from the subsequent Lems. 4.2 and 4.3.

Lemma 4.2. *The subsequence (X_{i_k}, Y_{i_k}) converging to (X^*, Y^*) in Thm. 4.1 are bounded and satisfy*

$$\lim_{i_k\to\infty} \nabla f_{X_{i_k}}(Y_{i_k-1}) = 0 \quad \textit{and} \quad \lim_{i_k\to\infty} \nabla f_{Y_{i_k}}(X_{i_k}) = 0 \tag{9}$$

Proof. First the boundedness of X_{i_k} and Y_{i_k} follows from the fact that they are convergent sequences. By the quadratic form of $f(X, Y)$ with respect to X and Y, the decrease of $f(X, Y)$ can be computed exactly in each iteration,

$$f(X_i, Y_{i-1}) = f(X_{i-1}, Y_{i-1}) - \frac{1}{2}\frac{|\langle \nabla f_{Y_{i-1}}(X_{i-1}), d_{x_{i-1}}\rangle|^2}{\|P_\Omega(d_{x_{i-1}}Y_{i-1})\|_F^2} \quad \text{and} \quad f(X_i, Y_i) = f(X_i, Y_{i-1}) - \frac{1}{2}\frac{|\langle \nabla f_{X_i}(Y_{i-1}), d_{y_{i-1}}\rangle|^2}{\|P_\Omega(X_i d_{y_{i-1}})\|_F^2}.$$

Summing the above two equations up from $i = 0$ to $i = \ell$ gives

$$f(X_\ell, Y_\ell) = f(X_0, Y_0) - \frac{1}{2}\sum_{i=0}^{\ell-1}\left\{\frac{|\langle \nabla f_{Y_i}(X_i), d_{x_i}\rangle|^2}{\|P_\Omega(d_{x_i}Y_i)\|_F^2} + \frac{|\langle \nabla f_{X_{i+1}}(Y_i), d_{y_i}\rangle|^2}{\|P_\Omega(X_{i+1}d_{y_i})\|_F^2}\right\}.$$

The nonnegativity of $f(X_\ell, Y_\ell)$ for all ℓ implies

$$\sum_{i=0}^{\ell-1}\left\{\frac{|\langle \nabla f_{Y_i}(X_i), d_{x_i}\rangle|^2}{\|P_\Omega(d_{x_i}Y_i)\|_F^2} + \frac{|\langle \nabla f_{X_{i+1}}(Y_i), d_{y_i}\rangle|^2}{\|P_\Omega(X_{i+1}d_{y_i})\|_F^2}\right\} < \infty.$$

Consequently,

$$\lim_{i\to\infty} \frac{|\langle \nabla f_{Y_i}(X_i), d_{x_i}\rangle|}{\|P_\Omega(d_{x_i}Y_i)\|_F} = 0 \quad \text{and} \quad \lim_{i\to\infty} \frac{|\langle \nabla f_{X_i}(Y_{i-1}), d_{y_{i-1}}\rangle|}{\|P_\Omega(X_i d_{y_{i-1}})\|_F} = 0. \tag{10}$$

Substituting the formulae for d_{x_i} and $d_{y_{i-1}}$ into (10) gives

$$\lim_{i\to\infty} \frac{|\langle \nabla f_{Y_i}(X_i), \nabla f_{Y_i}(X_i)(Y_iY_i^T)^{-1}\rangle|}{\|P_\Omega(\nabla f_{Y_i}(X_i)(Y_iY_i^T)^{-1}Y_i)\|_F} = 0 \quad \text{and} \quad \lim_{i\to\infty} \frac{|\langle \nabla f_{X_i}(Y_{i-1}), (X_i^TX_i)^{-1}\nabla f_{X_i}(Y_{i-1})\rangle|}{\|P_\Omega(X_i(X_i^TX_i)^{-1}\nabla f_{X_i}(Y_{i-1}))\|_F} = 0. \tag{11}$$

In particular, (11) remains true when i is replaced by the subsequence index i_k. From the assumption that $X_{i_k} \to X^*$, $Y_{i_k} \to Y^*$ and $X_{i_k},\ Y_{i_k},\ X^*,\ Y^*$ are nonsingular, we have

$$X_{i_k}^T X_{i_k} \to (X^*)^T X^*, \quad (X_{i_k}^T X_{i_k})^{-1} \to ((X^*)^T X^*)^{-1} \qquad \text{and} \qquad Y_{i_k} Y_{i_k}^T \to Y^*(Y^*)^T, \quad (Y_{i_k} Y_{i_k}^T)^{-1} \to (Y^*(Y^*)^T)^{-1}.$$

Let $C_{i_k} = (X_{i_k}^T X_{i_k})^{-1/2} \in \mathbb{R}^{r\times r}$ and $D_{i_k} = (Y_{i_k} Y_{i_k}^T)^{-1/2} \in \mathbb{R}^{r\times r}$. Then C_{i_k} and D_{i_k} are symmetric, positive definite and

$$C_{i_k} \to ((X^*)^T X^*)^{-1/2} \quad \text{and} \quad D_{i_k} \to (Y^*(Y^*)^T)^{-1/2}, \tag{12}$$

and the C_{i_k} and D_{i_k} are also bounded. The limits in (11) for the subsequence (X_{i_k}, Y_{i_k}) reduce to

$$\lim_{i_k\to\infty} \frac{|\langle \nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}, \nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}\rangle|}{\|P_\Omega(\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}^2 Y_{i_k})\|_F} = 0 \quad \text{and} \quad \lim_{i_k\to\infty} \frac{|\langle C_{i_k}\nabla f_{X_{i_k}}(Y_{i_k-1}), C_{i_k}\nabla f_{X_{i_k}}(Y_{i_k-1})\rangle|}{\|P_\Omega(X_i C_{i_k}^2 \nabla f_{X_i}(Y_{i-1})\|_F} = 0. \tag{13}$$

The boundedness of $X_{i_k},\ Y_{i_k},\ C_{i_k}$ and D_{i_k} implies

$$\begin{aligned}
\|P_\Omega(\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}^2 Y_{i_k})\|_F &\le \|\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}^2 Y_{i_k}\|_F \le \|\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}\|_F \cdot \|D_{i_k}\| \cdot \|Y_{i_k}\| \le C\|\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}\|_F && (14)\\
\|P_\Omega(X_i C_{i_k}^2 \nabla f_{X_i}(Y_{i_k-1})\|_F &\le \|X_i C_{i_k}^2 \nabla f_{X_i}(Y_{i_k-1})\|_F \le \|X_{i_k}\|_F \cdot \|C_{i_k}\|_F \cdot \|C_{i_k}\nabla f_{X_{i_k}}(Y_{i_k-1})\|_F \le C\|C_{i_k}\nabla f_{X_{i_k}}(Y_{i_k-1})\|_F && (15)
\end{aligned}$$

for some $C > 0$. Combining (13), (14) and (15) together gives

$$\lim_{i_k\to\infty} \|\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}\|_F = 0 \quad \text{and} \quad \lim_{i_k\to\infty} \|C_{i_k}\nabla f_{X_{i_k}}(Y_{i_k-1})\|_F = 0. \tag{16}$$

Therefore

$$\begin{aligned}
\lim_{i_k\to\infty} \|\nabla f_{Y_{i_k}}(X_{i_k})\|_F &= \lim_{i_k\to\infty} \|\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}D_{i_k}^{-1}\|_F \le \lim_{i_k\to\infty} \|\nabla f_{Y_{i_k}}(X_{i_k})D_{i_k}\|_F \cdot \|D_{i_k}^{-1}\|_F = 0, && (17)\\
\lim_{i_k\to\infty} \|\nabla f_{X_{i_k}}(Y_{i_k-1})\|_F &= \lim_{i_k\to\infty} \|C_{i_k}^{-1}C_{i_k}\nabla f_{X_{i_k}}(Y_{i_k-1})\|_F \le \lim_{i_k\to\infty} \|C_{i_k}^{-1}\|_F \|C_{i_k}\nabla f_{X_{i_k}}(Y_{i_k-1})\|_F = 0, && (18)
\end{aligned}$$

where we use the fact $\lim_{i_k\to\infty} \|D_{i_k}^{-1}\|_F = \|(Y^*(Y^*)^T)^{1/2}\|_F < \infty$ and $\lim_{i_k\to\infty} \|C_{i_k}^{-1}\|_F = \|((X^*)^T X^*)^{1/2}\|_F < \infty$. □

Lemma 4.3. *The subsequence (X_{i_k}, Y_{i_k}) in Thm. 4.1 satisfy*

$$\lim_{i_k\to\infty} \nabla f_{X_{i_k}}(Y_{i_k}) = 0. \tag{19}$$

Proof. The difference between $\nabla f_{X_{i_k}}(Y_{i_k})$ and $\nabla f_{X_{i_k}}(Y_{i_k-1})$ is

$$\begin{aligned}
\nabla f_{X_{i_k}}(Y_{i_k}) - \nabla f_{X_{i_k}}(Y_{i_k-1}) &= -X_{i_k}^T(P_\Omega(Z^0) - P_\Omega(X_{i_k}Y_{i_k})) + X_{i_k}^T(P_\Omega(Z^0) - P_\Omega(X_{i_k}Y_{i_k-1}))\\
&= X_{i_k}^T P_\Omega(X_{i_k}(Y_{i_k} - Y_{i_k-1})) = X_{i_k}^T t_{y_{i_k-1}} P_\Omega(X_{i_k} d_{y_{i_k-1}})\\
&= -X_{i_k}^T \frac{\langle \nabla f_{X_{i_k}}(Y_{i_k-1}), d_{y_{i_k-1}}\rangle}{\|P_\Omega(X_{i_k} d_{y_{i_k-1}})\|_F^2} P_\Omega(X_{i_k} d_{y_{i_k-1}}),
\end{aligned}$$

where the third equation follows from $Y_{i_k} = Y_{i_k-1} + t_{y_{i_k-1}} d_{y_{i_k-1}}$. Hence,

$$\|\nabla f_{X_{i_k}}(Y_{i_k}) - \nabla f_{X_{i_k}}(Y_{i_k-1})\|_F \le \|X_{i_k}\|_F \frac{|\langle \nabla f_{X_{i_k}}(Y_{i_k-1}), d_{y_{i_k-1}}\rangle|}{\|P_\Omega(X_{i_k} d_{y_{i_k-1}})\|_F} \le C \cdot \frac{|\langle \nabla f_{X_{i_k}}(Y_{i_k-1}), d_{y_{i_k-1}}\rangle|}{\|P_\Omega(X_{i_k} d_{y_{i_k-1}})\|_F} \to 0, \tag{20}$$

where the second inequality follows from the boundedness of X_{i_k} and the limit follows from (10). Combining (20) and the left limit of (9) gives $\lim_{i_k\to\infty} \nabla f_{X_{i_k}}(Y_{i_k}) = 0$. □

5. Numerical experiments

In this section, we present the empirical performance of our gradient based algorithms as well as related algorithms. ASD and ScaledASD are implemented in Matlab with two subroutines written in C to take advantage of the sparse structure. Other tested algorithms include PowerFactorization (PF) [12], LMaFit [24], LRGeomCG [23], and ScGrassMC [20], which are all downloaded from the authors' websites and where applicable use the aforementioned C subroutines. We compare these algorithms on standard test images in Sec. 5.1 and random low rank matrices in Sec. 5.2.

5.1. Image inpainting

In this section, we demonstrate the performance of the proposed algorithms on image inpainting problems. Image inpainting concerns filling in the unknown pixels of an image from an incomplete set of observed entries. All the aforementioned algorithms are tested on three standard 512×512 grayscale test images (Boat, Barbara, and Lena), each projected to the nearest rank 50 image in Frobenius norm so as to allow recovery to arbitrary precision. Two sampling schemes are considered, 1) random sampling where 35% pixels of the low rank image were sampled uniformly at random, and 2) cross mask where 6.9% pixels of of the image were masked in a non-random cross pattern centred in each image. The relative residual tolerance was set to 10^{-5}. All algorithms were provided with the true rank of the image except LMaFit for random sampling which used warm starting strategies proposed by its authors in [24] where the hand tuned parameters chosen to give the best performance on the images tested were their increasing strategy with initial rank set to 25 and maximum rank set to 60. The simulation was conducted on a computer with Intel Xeon E5-2643 CPUs @ 3.30 GHz and 64GB memory running linux and executed from Matlab R2013a.

The reconstructed images for the Boat test images and each tested algorithm can be found in [22]. Here we only present the relative residual plotted against the computational time, see Fig. 1. For random sampling, Fig. 1 (a,b,c) show that ScaledASD and ScGrassMC have rapid initial decrease in the residual, and that LRGeomCG has the fastest asymptotic rate. For moderate accuracy ScaledASD and ScGrassMC require the least time for random sampling of the tested natural images, while for high accuracy LRGeomCG is preferable. It should be noted that the completed images are visually indistinguishable for relative residuals of about 10^{-2}, which is well before the fast asymptotic rate of LRGeomCG begins. Alternatively, for the cross mask, Fig. 1 (d,e,f) show that ScaledASD, PowerFactorization, and LMaFit are preferable throughout, though the fast asymptotic rate of LRGeomCG suggests it would be superior for a relative tolerance below 10^{-5}. Only ScaledASD is preferred for both random sampling and the cross mask, suggesting its usage for moderate accuracy, and LRGeomCG for higher accuracy.

5.2. Random matrix completion problems

In this section we conduct tests of randomly drawn rank r matrices using the model $Z^0 = XY$, where $X \in \mathbb{R}^{m\times r}$ and $Y \in \mathbb{R}^{r\times n}$ with X and Y having their entries drawn i.i.d. from the normal distribution $\mathcal{N}(0,1)$. A random subset Ω of p entries in Z^0 are sampled uniformly at random. For conciseness, the tests presented in this section consider square matrices as is typical in the literature.

5.2.1. Largest recoverable rank

Of central importance in matrix completion is what is the largest recoverable rank given (p,m,n), or equivalently, given (r,m,n) how many of the entries are needed in order to reliably recover a rank r matrix. To quantify the optimality of such statements we use the phase transition framework which for matrix completion is defined through the *undersampling* and *oversampling ratios*:

$$\delta = \frac{p}{mn} \quad \text{and} \quad \rho = \frac{d_r}{p}, \tag{21}$$

where p is the number of sampled entries, and $d_r = r(m+n-r)$ is the degrees of freedom in a $m\times n$ rank r matrix. The region of greatest importance is severe undersampling, for $\delta \ll 1$, and we restrict our tests to the two values $\delta \in \{0.05, 0.1\}$, which also allows us testing of large matrices[1]. Due to the high computational cost of accurately

[1]Tests were conducted on the IRIDIS HPC facility provided by the e-Infrastructure South Centre for Innovation which is composed of Linux nodes composed of two 6-core 2.4GHz Intel Westmere processors and approximately 22GB of usable memory per node. The data presented required 4.5 months of CPU time.

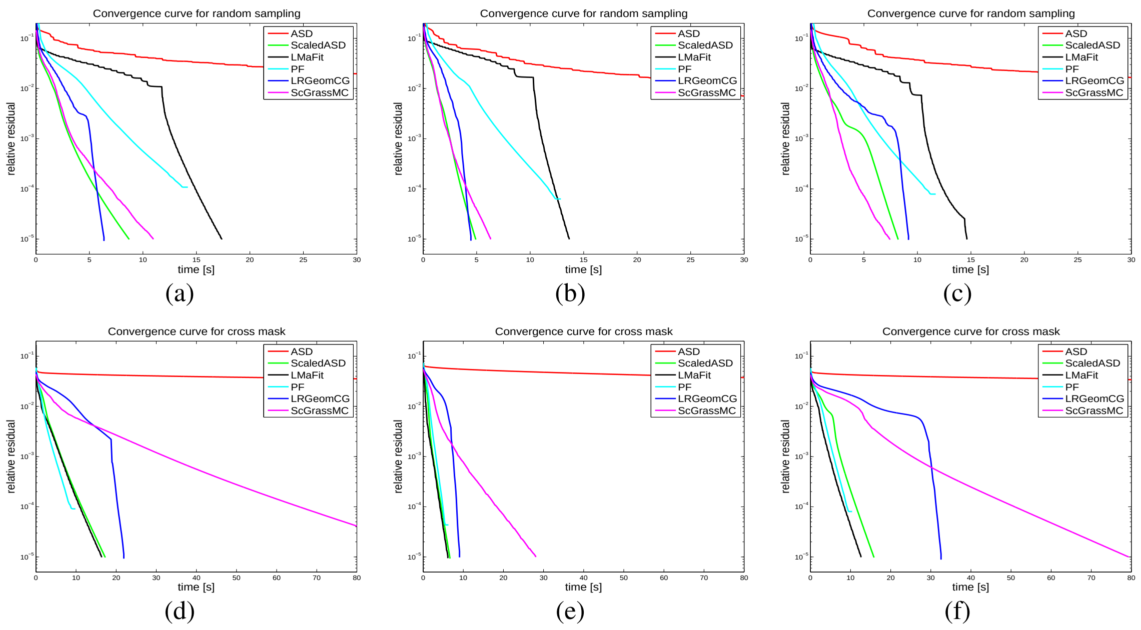

Figure 1: Relative residual as function of computational time for image recovery testing on "Boat" (left), "Barbara"(middle) and "Lena"(right). Top panels: random sampling; bottom panels: cross mask.

computing an algorithm's phase transition we limit the testing here to four of the most efficient algorithms ASD, ScaledASD, LMaFit, and LRGeomCG. ASD, ScaledASD and LRGeomCG were provided with the true rank of the matrix, while the decreasing strategy with the initial rank set to $\lfloor 1.25r \rfloor$ was used in LMaFit as suggested in [24]. In these tests an algorithm is considered to have successfully recovered the sampled matrix Z^0 if it returned a matrix Z^{out} that is within 10^{-3} of Z^0 in the relative Frobenius norm, that is if $rel.err := \|Z^{\text{out}} - Z^0\|_F / \|Z^0\|_F \leq 10^{-3}$. For each triple (m, n, p), we started from a sufficiently small rank r so that the algorithm could successfully recover each of the sampled matrices in 100 random tests. The rank is then increased by 1 until it was large enough that the algorithm failed to recovery any of 100 tests to within the prescribed relative accuracy. We refer to the largest rank for which the algorithm succeeded in each of the 100 tests as r_{min}, and the smallest rank for which the algorithm failed all the 100 tests as r_{max}. The exact values of r_{min}, r_{max} and associated ρ_{min}, ρ_{max} are listed in Tab. 1.

Table 1 shows that ASD, ScaledASD and LGeomCG are able to recovering a rank r matrix from only $C \cdot r(m+n-r)$ measurements of its entries with C being only slightly larger than 1. The ability of recovering a low rank matrix from almost near the minimum number of measurements was firstly reported for NIHT by the authors in [21]. In contrast, LMaFit has a substantially lower phase transition. Table 1 shows that these observations are consistent for different, moderately large size matrices.

5.2.2. Recovery time

In this section we evaluate the recovery time for the aforementioned algorithms when applied to the previously described randomly generated rank r matrices. The relative error, number of iterations, and computational time reported in this section are average results of 10 random tests. All algorithms were supplied with the true rank, except LMaFit which used the decreasing strategy with the initial rank $\lfloor 1.25r \rfloor$ as advocated in [24]. The tolerance for relative residual, $\|P_\Omega(Z^{\text{out}} - Z^0)\|_2 / \|P_\Omega(Z^0)\|_2$, was set to 10^{-5} for each algorithm and all parameters were set to their default values. The tests were performed on the same computer as described in Sec. 5.1.

The performance of ASD, ScaledASD, and PF are compared in Tab. 2 for medium size matrices of $m = n = 1000$ with undersampling ratio $\delta \in \{0.1, 0.3, 0.5\}$ and the rank of the matrix is taken so that $p/d_r \approx 2.0$ and 1.27. Table 2 shows that PF takes many fewer iterations than ASD and ScaledASD; however, both ASD and ScaledASD can require much less computational time than PF due to their lower per iteration computational complexity. ScaledASD is observed to take fewer number of iterations than ASD and is typically faster. The advantage of ScaledASD over ASD

Table 1: Phase transition table for ASD, ScaledASD, LMaFit, and LRGeomCG. For each (m, n, p) with $p = \delta \cdot mn$, if $r \leq r_{min}$, the algorithm was observed to recover each of the 100 randomly drawn $m \times n$ matrices of rank r and failed to recover each of the randomly drawn matrices if $r \geq r_{max}$. The oversampling ratios ρ_{min} and ρ_{max} are computed using r_{min} and r_{max} by (21).

Configuration		ASD & ScaledASD				LRGeomCG				LMaFit			
m = n	δ	r_{min}	r_{max}	ρ_{min}	ρ_{max}	r_{min}	r_{max}	ρ_{min}	ρ_{max}	r_{min}	r_{max}	ρ_{min}	ρ_{max}
1000	0.05	18	23	0.71	0.91	18	24	0.71	0.95	12	22	0.48	0.87
1000	0.10	43	48	0.84	0.94	44	48	0.86	0.94	30	44	0.59	0.86
2000	0.05	44	47	0.87	0.93	44	47	0.87	0.93	25	41	0.50	0.81
2000	0.10	92	95	0.90	0.93	92	96	0.90	0.94	54	84	0.53	0.82
4000	0.05	91	94	0.90	0.93	92	95	0.91	0.94	65	84	0.64	0.83
4000	0.10	186	190	0.91	0.93	188	191	0.92	0.93	141	172	0.69	0.84

increases as when $\delta = p/mn$ increases for p/d_r being approximately fixed, because the more entries of a matrix are known, the more ScaledASD behaves like a second order method. In particular, if all the entries have been observed, ASD and ScaledASD are alternating gradient descent and Newton method for (8) respectively.

Table 2: Average computational time (seconds) and average number of iterations of ASD, ScaledASD and PF over ten random rank r matrices per (p, m, n, r) tuple for $m = 1000, n = 1000$, $\delta \in \{0.1, 0.3, 0.5\}$.

p/mn	p/d_r	ASD			ScaledASD			PF		
		rel.err	iter	time	rel.err	iter	time	rel.err	iter	time
0.1	2.05	3.5e-5	103	1.5	3.5e-5	97	**1.5**	2.6e-5	29	11.0
	1.28	9.0e-5	582	14.1	9.1e-5	533	**13.3**	8.1e-5	141	85.4
0.3	2.05	2.6e-5	63	8.8	2.6e-5	48	**6.8**	2.1e-5	20	14.0
	1.28	7.0e-5	373	88.7	7.0e-5	252	**61.6**	6.7e-5	93	110
0.5	2.05	2.1e-5	64	26.3	2.3e-5	33	**13.7**	1.6e-5	16	16.5
	1.28	5.8e-5	361	255	6.0e-5	155	**111**	5.7e-5	65	124

Next we compare the algorithms on larger problem sizes, with the exception of PF due to its its prohibitive computational time, and show their performance dependence on additive noise. Tests with additive noise have the sampled entries $P_\Omega(Z^0)$ corrupted by the vector $e = \epsilon \cdot \|P_\Omega(Z^0)\|_2 \cdot w/\|w\|_2$, where the entries of w are i.i.d standard Gaussian random variables, and ϵ is referred to as the noise level. The tests conducted here consider ϵ to be either zero or 0.1. Tests are conducted for $m = n \in \{8000, 16000\}$, $r \in \{40, 80\}$, and $p/d_r \in \{3, 5\}$. The results are presented in Tabs. 3 and 4 for noise levels $\epsilon = 0$ and 0.1 respectively. Table 3 shows that for $\epsilon = 0$: ASD and ScaledASD consistently require less time than LMaFit, ScaledASD always requires the least time for $p/d_r = 5$ and LRGeomCG requires the least time for $p/d_r = 3$ where the problems become less well conditioned. Despite the efficiency of ScGrassMC for random sampling of the natural images in Fig. 1 (a,c,e), Tabs. 3 and 4 show that ScGrassMC consistently requires substantially greater computational time, form random rank r matrices, than the other algorithms tested. Tables 3 and 4 show the efficiency of ASD and ScaledASD when the random problems are solved to moderate accuracy. However it is worth noting that these randomly generated matrices are well-conditioned with high probability; as ASD and ScaledASD are both first order methods, it can be expected that they will suffer from slow asymptotic convergence for severely ill-conditioned matrices, in which case a higher order method such as LRGeomCG or ScGrassMC may be preferable, see [4, 20] for a discussion of this phenomenon.

6. Conclusion

We have proposed an alternating steepest method, ASD, and a scaled variant, ScaledASD, for matrix completion. Despite their simplicity, empirical evaluation of ASD and ScaledASD on both image in painting and random problems show ASD and ScaledASD to be competitive with other state-of-the-art matrix completion algorithms in terms of recoverable rank and overall computational time. In particular, their low per iteration computational complexity

Table 3: Average computational time (seconds) and average number of iterations of ASD, ScaledASD, LMaFit and LRGeomCG over ten random rank r matrices per (p, m, n, r) tuple for $m = n \in \{4000, 8000, 16000\}$, $r \in \{40, 80\}$ and $p/d_r \in \{3, 5\}$. The noise level ϵ is equal to 0.

r	40						80					
p/d_r	3			5			3			5		
	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time
$m = n$	8000											
ASD	2.1e-5	43	24.6	1.4e-5	23	21.1	1.9e-5	37	80.8	1.4e-5	20	70.9
ScaledASD	2.1e-5	43	24.4	1.4e-5	23	**20.8**	1.9e-5	36	78.6	1.4e-5	20	**68.6**
LMaFit	1.9e-5	73	29.2	1.5e-5	38	24.6	1.9e-5	62	94.0	1.3e-5	35	87.0
LRGeomCG	1.7e-5	23	**20.4**	8.1e-6	16	22.2	1.3e-5	22	**74.1**	1.2e-5	15	77.4
ScGrassMC	1.4e-5	29	179	8.9e-6	16	265	1.7e-5	23	2330	9.5e-6	15	3445
$m = n$	16000											
ASD	2.1e-5	44	61.4	1.4e-5	23	50.4	1.9e-5	38	201	1.1e-5	21	169
ScaledASD	2.0e-5	43	56.4	1.4e-5	23	**47.8**	2.0e-5	37	186	1.4e-5	20	**159**
LMaFit	2.2e-5	81	77.5	1.3e-5	41	62.6	1.7e-5	69	242	1.4e-5	37	210
LRGeomCG	1.6e-5	24	**47.5**	1.0e-5	16	49.9	1.4e-5	22	**167**	9.7e-6	15	178
ScGrassMC	1.5e-5	25	495	1.0e-5	16	704	1.4e-5	23	6063	1.0e-5	16	8681

Table 4: Average computational time (seconds) and average number of iterations of ASD, ScaledASD, LMaFit and LRGeomCG over ten random rank r matrices per (p, m, n, r) tuple for $m = n \in \{4000, 8000, 16000\}$, $r \in \{40, 80\}$ and $p/d_r \in \{3, 5\}$. The noise level ϵ is equal to 0.1.

r	40						80					
p/d_r	3			5			3			5		
	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time	rel.err	iter	time
$m = n$	8000											
ASD	7.1e-2	21	**12.3**	5.0e-2	19	17.9	7.0e-2	21	46.2	5.0e-2	19	67.1
ScaledASD	7.1e-2	21	12.3	5.0e-2	19	18.0	7.0e-2	21	46.5	5.0e-2	19	67.5
LMaFit	7.1e-2	34	14.8	5.0e-2	22	15.7	7.0e-2	31	49.7	5.0e-2	20	**52.7**
LRGeomCG	7.1e-2	14	12.7	5.0e-2	11	**15.5**	7.0e-2	13	**44.7**	5.0e-2	11	60.8
ScGrassMC	7.1e-2	16	171	5.0e-2	10	264	7.0e-2	14	2337	5.0e-2	10	3623
$m = n$	16000											
ASD	7.1e-2	21	29.8	5.0e-2	20	41.9	7.1e-2	21	108	5.0e-2	19	157
ScaledASD	7.1e-2	21	**28.2**	5.0e-2	20	42.3	7.1e-2	21	**106**	5.0e-2	19	152
LMaFit	7.3e-2	29	31.1	5.0e-2	26	42.4	7.1e-2	32	119	5.0e-2	22	135
LRGeomCG	7.1e-2	15	29.4	5.0e-2	11	**35.7**	7.1e-2	14	106	5.0e-2	11	**133**
ScGrassMC	7.1e-2	16	651	5.0e-2	11	757	7.1e-2	15	6285	5.0e-2	10	9227

Table 5: Average relative error of ASD over ten random tests in reconstructing rank r matrices of size $m = n = 1000$ when the rank provided is k, where $k = cr$ with $c \in \{1/2, 4/5, 5/4, 2\}$. The ratio compares the reconstruction error over the error when truncating the matrix at rank k.

Sampling ratio	True rank	$k = r/2$		$k = 4r/5$		$k = 5r/4$		$k = 2r$	
		rel.err	ratio	rel.err	ratio	rel.err	ratio	rel.err	ratio
$\delta = 0.2$	$r = 40$ ($\rho = 0.39$)	0.69	1.12	0.432	1.21	3.27e-5	-	2.3e-2	-
	$r = 80$ ($\rho = 0.77$)	0.834	1.45	0.669	2.09	1.4e-2	-	0.673	-
$\delta = 0.5$	$r = 100$ ($\rho = 0.38$)	0.603	1.08	0.347	1.136	2.08e-5	-	6.4e-5	-
	$r = 200$ ($\rho = 0.72$)	0.615	1.245	0.381	1.54	7.1e-5	-	0.537	-

makes ASD and ScaledASD efficient for large size problems, especially when computing the solutions to moderate accuracy. In this paper, ASD and ScaledASD are implemented for fixed rank problems. When the rank of the matrix in not known a priori, it can be estimated, for example, based on the gap of the singular values of the trimmed partial observed matrix, see [15, Fig. 1]. The effect of applying ASD with the rank under or overestimated is explored in Tab. 5 where ASD is applied with rank k where the true rank is r. Tab. 5 reports the relative error of the reconstruction from ASD for four ratios of $k/r = \{1/2, 4/5, 5/4, 2\}$ and when $k < r$ the ratio of the reconstruction error over the best rank k approximation is listed. The table shows that when $k < r$, ASD returns a matrix that is comparable with the best rank r approximation matrix with the ratio $\lesssim 2$. When $k > r$ and $\rho = \frac{k(m+n-k)}{p} < 1$, ASD returns a matrix that is close to the rank r matrix. However, the reconstruction error increases with k/r due to the finite maximum number of iterations imposed.

Acknowledgment

The authors thank the reviewers for their useful suggestions which have improved this manuscript.

References

[1] Y. Amit, M. Fink, N. Srebro, and S. Ullman. Uncovering shared structures in multiclass classification. *in Proceedings of the 24th international conference on machine learning, ACM*, pages 17–24, 2007.

[2] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. *Advances in Neural Information Processing Systems*, 19:41–48, 2007.

[3] J. Blanchard, J. Tanner, and K. Wei. CGIHT: Conjugate gradient iterative hard thresholing for compressed sensing and matrix completion. Numerical analysis group preprint 14/08, University of Oxford, 2014.

[4] N. Boumal and P.-A. Absil. RTRMC: A Riemannian trust-region method for low-rank matrix completion. In J. Shawe-Taylor, R.S. Zemel, P. Bartlett, F.C.N. Pereira, and K.Q. Weinberger, editors, *Advances in Neural Inf. Processing Systems 24 (NIPS)*, pages 406–414. 2011.

[5] J. F. Cai, E. J. Candès, and Z. W. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[6] E. J. Candès and B. Recht. Exact matrix completion via convex optimization. *Foundations of Computational Mathematics*, 9(6):717–772, 2009.

[7] E. J. Candès and T. Tao. The power of convex relaxation: Near-optimal matrix completion. *IEEE Transactions on Information Theory*, 56(5):2053–1080, 2009.

[8] L. Eldén. *Matrix methods in data mining and pattern recognization*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2007.

[9] R. Escalante and M. Raydan. *Alternating Projection Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2011.

[10] M. Fazel. Matrix rank minimization with applications. Ph. D. dissertation, Stanford University, 2002.

[11] L. Grippo and M. Sciandrone. Gloabally convergence block-coordinate techniques for unconstrained optimization. *Optimization Methods & Software*, 10(4):587–637, 1999.

[12] J. P. Haldar and D. Hernando. Rank-constrained solutions to linear matrix equations using PowerFactorization. *IEEE Signal Processing Letters*, 16(7):584–587, 2009.

[13] N. J. A. Harvey, D. R. Karger, and S. Yekhanin. The complexity of matrix completion. *SODA 2006, Proceedings of the seventeenth annual ACM-SIAM symposium on discrete algorithms*, pages 1103–1111, 2006.

[14] P. Jain, R. Meka, and I. Dhillon. Guaranteed rank minimization via singular value projection. *Proceedings of the Neural Information Processing Systems Conf. (NIPS)*, pages 937–945, 2010.

[15] R. H. Keshavan, A. Montanari, and S. Oh. Matrix completion from a few entries. *IEEE Transactions on Information Theory*, 56(6):2980–2998, 2010.

[16] A. Kyrillidis and V. Cevher. Matrix recipes for hard thresholding methods. *Journal of Mathematical Imaging and Vision*, 48(2):235–265, 2014.

[17] Z. Liu and L. Vandenberghe. Interior-point method for nuclear norm approximation with application to system identification. *SIAM Journal on Matrix Analysis and Applications*, 31:1235–1256, 2009.

[18] B. Mishra, K. A. Apuroop, and R. Sepulchre. A Riemannian geometry for low-rank matrix completion. arXiv:1306.2672, 2013.

[19] B. Mishra, G. Meyer, S. Bonnabel, and R. Sepulchre. Fixed-rank matrix factorizations and Riemannian low-rank optimization. *Computational Statistics*, 2013. doi:10.1007/s00180-013-0464-z.

[20] T. Ngo and Y. Saad. Scaled gradients on grassmann manifolds for matrix completion. In *Advances in Neural Information Processing Systems(NIPS)*. 2012.

[21] J. Tanner and K. Wei. Normalized iterative hard thresholding for matrix completion. *SIAM J. Sci. Comput.*, 35(5):S104–S125, 2013.

[22] J. Tanner and K. Wei. Low rank matrix completion by alternating steepest descent methods. Numerical Analysis Group preprint (ID Code: 1838), University of Oxford, 2014.

[23] B. Vandereycken. Low rank matrix completion by Riemannian optimization. *SIAM Journal on Optimization*, 23(2):1214–1236, 2013.

[24] Z. Wen, W. Yin, and Y. Zhang. Solving a low-rank factorization model for matrix completion by a non-linear successive over-relaxation algorithm. *Mathematical Programming Computation*, 4:333–361, 2012.

[25] Y. Xu and W. Yin. A block coordinate descent method for regularized multi-convex optimization with applications to nonnegative tensor factorization and completion. *SIAM J. Imaging Sciences*, 6(3):1758–1789, 2013.