Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)

Accelerated Inexact Soft-Impute for Fast Large-Scale Matrix Completion

Quanming Yao   **James T. Kwok**
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{qyaoaa, jamesk}@cse.ust.hk

Abstract

Matrix factorization tries to recover a low-rank matrix from limited observations. A state-of-the-art algorithm is the Soft-Impute, which exploits a special "sparse plus low-rank" structure of the matrix iterates to allow efficient SVD in each iteration. Though Soft-Impute is also a proximal gradient algorithm, it is generally believed that acceleration techniques are not useful and will destroy the special structure. In this paper, we show that Soft-Impute can indeed be accelerated without compromising the "sparse plus low-rank" structure. To further reduce the per-iteration time complexity, we propose an approximate singular value thresholding scheme based on the power method. Theoretical analysis shows that the proposed algorithm enjoys the fast $O(1/T^2)$ convergence rate of accelerated proximal gradient algorithms. Extensive experiments on both synthetic and large recommendation data sets show that the proposed algorithm is much faster than Soft-Impute and other state-of-the-art matrix completion algorithms.

1 Introduction

In many applications, the data are stored in a matrix. Given a partially observed matrix O, matrix factorization attempts to recover a low-rank matrix X that best approximates O on the observed entries [Candès and Recht, 2009]. Matrix factorization has been widely used in a variety of domains. Examples include collaborative filtering for recommender systems [Mazumder *et al.*, 2010], link prediction for social networks [Kim and Leskovec, 2011], image inpainting [Liu *et al.*, 2013], and multilabel learning [Cabral *et al.*, 2011].

However, direct minimization of the matrix rank is NP-hard, and the nuclear norm (which is the sum of singular values) is often used instead. Mathematically, let $O \in \mathbb{R}^{m \times n}$ (where, without loss of generality, we assume that $m \geq n$), and the positions of the observed entries be indicated by $\Omega \in \{0,1\}^{m \times n}$, such that $\Omega_{ij} = 1$ if O_{ij} is observed, and 0 otherwise. Matrix factorization is formulated as the following optimization problem

$$\min_X \frac{1}{2}\|P_\Omega(X - O)\|_F^2 + \lambda\|X\|_*, \qquad (1)$$

where $[P_\Omega(A)]_{ij} = A_{ij}$ if $\Omega_{ij} = 1$, and 0 otherwise; and $\|X\|_*$ is the nuclear norm of X. It is known that the nuclear norm is the tightest convex lower bound of the rank [Recht *et al.*, 2010]. Besides, though the nuclear norm is only a surrogate, there are theoretical guarantees that the underlying matrix can be recovered [Candès and Recht, 2009].

Computationally, though the nuclear norm is nonsmooth, problem (1) can be solved by various optimization tools. An early attempt is based on reformulating (1) as a semidefinite program (SDP) [Candès and Recht, 2009]. However, SDP solvers have large time and space complexities, and are only suitable for small data sets. For large-scale matrix completion, Cai *et al.* [2010] pioneered the use of first-order methods and proposed the singular value thresholding (SVT) algorithm. However, a singular value decomposition (SVD) is required in each SVT iteration. This takes $O(mn^2)$ time and can be computationally expensive. In [Toh and Yun, 2010], this is reduced to a partial SVD by computing only the leading singular values/vectors using PROPACK (a variant of the Lanczos algorithm) [Larsen, 1998]. Another major breakthrough is made by the Soft-Impute algorithm [Mazumder *et al.*, 2010], which utilizes a special "sparse plus low-rank" structure associated with the SVT to efficiently compute the SVD. Empirically, this allows Soft-Impute to perform matrix completion on the entire Netflix data set with a reasonable time.

The SVT algorithm can also be viewed as a proximal gradient algorithm [Tibshirani, 2010]. Hence, it converges with a $O(1/T)$ rate, where T is the number of iterations [Beck and Teboulle, 2009; Nesterov, 2013]. Later, this is further "accelerated", and the convergence rate is improved to $O(1/T^2)$ [Ji and Ye, 2009; Toh and Yun, 2010]. However, Tibshirani [2010] suggested that this is not useful, as the special "sparse plus low-rank" structure crucial to the efficiency of Soft-Impute no longer exist. In other words, the gain in convergence rate is more than compensated by the increase in per-iteration time complexity.

In this paper, we show that accelerating Soft-Impute is indeed possible while simultaneously preserving the "sparse plus low-rank" structure. Moreover, instead of using PROPACK to compute the (exact) partial SVD as in [Toh and Yun, 2010], we propose to use the power method [Halko *et al.*, 2011] to obtain only an approximation of the dominant singular subspace. This is more efficient than PROPACK and also allows warm-start, which is particularly useful because of the iterative nature of Soft-Impute. Since the SVT obtained is then only approximate, we use recent results in proximal gradient algorithms [Schmidt *et al.*, 2011] to show that convergence can still be as fast as performing exact SVT in each iteration. Hence, the resultant algorithm has low per-iteration complexity and fast $O(1/T^2)$ convergence rate.

The rest of the paper is organized as follows. Section 2 provides a brief review on the related work. The proposed accelerated inexact Soft-Impute algorithm is described in Section 3. Experimental results are presented in Section 4, and the last section gives some concluding remarks.

Notation

In the sequel, the transpose of vector/ matrix is denoted by the superscript $^\top$. For a vector x, $\|x\|_1 = \sum_i |x_i|$ is its ℓ_1-norm, and $\|x\| = \sqrt{\sum_i x_i^2}$ its ℓ_2-norm. For a matrix X, $\sigma_1 \geq \sigma_2 \geq \dots$ are its singular values, $\text{tr}(X) = \sum_i X_{ii}$ is its trace, $\|X\|_1 = \sum_{i,j} |X_{ij}|$, $\|X\|_F = \text{tr}(X^\top X)$ is the Frobenius norm, and $\|X\|_* = \sum_i \sigma_i$ the nuclear norm. Moreover, I denotes the identity matrix. For a function f, we let f' be a subgradient in the subdifferential $\partial f(x) = \{g \mid f(y) \geq f(x) + g^T(y-x), \ \forall y\}$. When f is differentiable, we use ∇f for its gradient. Finally, we use $\Omega^\perp$ to denote the complement of a set Ω. For a matrix A, $[P_\Omega(A)]_{ij} = \begin{cases} A_{ij} & \Omega_{ij} = 1 \\ 0 & \text{otherwise} \end{cases}$, and $[P_\Omega^\perp(A)]_{ij} = \begin{cases} 0 & \Omega_{ij} = 1 \\ A_{ij} & \text{otherwise} \end{cases}$.

2 Related Work

2.1 Proximal Gradient Algorithms

Consider minimizing composite functions of the form:

$$F(x) \equiv f(x) + g(x), \tag{2}$$

where f, g are convex, f is smooth but g is possibly nonsmooth. The proximal gradient algorithm [Beck and Teboulle, 2009; Nesterov, 2013; Parikh and Boyd, 2014] generates a sequence of estimates $\{x_t\}$ as

$$x_{t+1} = \text{prox}_{\mu g}(z_t) \equiv \arg\min_x \frac{1}{2}\|x - z_t\|^2 + \mu g(x), \tag{3}$$

where

$$z_t = x_t - \mu \nabla f(x_t), \tag{4}$$

and $\text{prox}_{\mu g}(\cdot)$ is called the proximal operator. When f has ρ-Lipschitz continuous gradient (i.e., $\|\nabla f(x_1) - \nabla f(x_2)\| \leq \rho\|x_1 - x_2\|$) and a fixed stepsize $\mu \leq 1/\rho$ is used, this algorithm converges with rate $O(1/T)$, where T is the number of iterations [Parikh and Boyd, 2014]. Moreover, it can be accelerated by replacing (4) with

$$y_t = (1 + \theta_t)x_t - \theta_t x_{t-1}, \quad z_t = y_t - \mu \nabla f(y_t). \tag{5}$$

Here, several choices for θ_t can be used. The resultant accelerated proximal gradient (APG) algorithm converges with the optimal $O(1/T^2)$ rate [Nesterov, 2013].

In the sequel, as our focus is on matrix completion, the variable x in (2) is a matrix X.

2.2 Soft-Impute

Soft-Impute is a state-of-the-art algorithm for large-scale matrix completion [Mazumder *et al.*, 2010]. At iteration t, the missing values in O are filled in as

$$Z_t = P_\Omega(O) + P_\Omega^\perp(X_t) = P_\Omega(O - X_t) + X_t, \tag{6}$$

where X_t is the current iterate. X_{t+1} is then generated as

$$X_{t+1} = \text{SVT}_\lambda(Z_t), \tag{7}$$

where $\text{SVT}_\lambda(\cdot)$ is the singular value thresholding (SVT) operator defined as follows.

Lemma 2.1 ([Cai *et al.*, 2010]). *Let the SVD of a matrix Z be $U\Sigma V^\top$. The solution to the optimization problem*

$$\min_X \frac{1}{2}\|X - Z\|_F^2 + \lambda\|X\|_*$$

is a low-rank matrix given by

$$SVT_\lambda(Z) \equiv U(\Sigma - \lambda I)_+ V^\top, \tag{8}$$

where $[(A)_+]_{ij} = \max(A_{ij}, 0)$.

To compute X_{t+1} in (7), we thus need to first perform SVD on Z_t. In general, obtaining the rank-k SVD of a $m \times n$ matrix Z takes $O(mnk)$ time. Its most expensive steps are on computing matrix-vector multiplications of the form Zu and $v^\top Z$, where $u \in \mathbb{R}^n$ and $v \in \mathbb{R}^m$.

To make Soft-Impute efficient, an important observation by Mazumder *et al.* is that Z_t in (6) has a special "sparse plus low-rank" structure, namely that $P_\Omega(O - X_t)$ is sparse and X_t is low-rank. Let the rank of X_t be r, and its SVD be $U_t \Sigma_t V_t^\top$. For any $u \in \mathbb{R}^n$, $Z_t u$ can then be computed as

$$Z_t u = P_\Omega(O - X_t)u + U_t \Sigma_t (V_t^\top u). \tag{9}$$

Constructing $P_\Omega(O - X_t)$ takes $O(r\|\Omega\|_1)$ time while computing $P_\Omega(O - X_t)u$ takes $O(\|\Omega\|_1)$ time, and computing $U_t \Sigma_t (V_t^\top u)$ takes $O((m+n)r)$ time. Thus, to obtain the rank-k SVD of Z_t, Soft-Impute needs only $O((r+k)\|\Omega\|_1 + rk(m+n))$ time. Moreover, its convergence rate is $O(1/T)$.

In (1), take $f(X) \equiv \frac{1}{2}\|P_\Omega(X - O)\|_F^2$ and $g(X) \equiv \lambda\|X\|_*$. Note that $\nabla f(X) = P_\Omega(X - O)$, and $\|\nabla f(X_1) - \nabla f(X_2)\|_F^2 = \|P_\Omega(X_1 - X_2)\|_F^2 \leq \|X_1 - X_2\|_F^2$. Hence, f is convex and has 1-Lipschitz-continuous gradient (i.e., $\rho = 1$); while g is also convex but nonsmooth. It is obvious that Z_t in (4) is the same as that in (6). Using $\mu = 1/\rho = 1$, it can be easily shown that $\text{prox}_{\mu g}(Z_t) = \text{prox}_g(Z_t)$ in (3) is indeed equal to $\text{SVT}_\lambda(Z_t)$. Hence, interestingly, Soft-Impute is also a proximal gradient algorithm [Tibshirani, 2010].

3 Accelerated Inexact Soft-Impute

Since Soft-Impute is a proximal gradient algorithm, it is natural to consider accelerating it with the schemes in Section 2.1. However, Tibshirani [2010] suggested that this is not useful, as the special "sparse plus low-rank" structure crucial to the efficiency of Soft-Impute will be lost. In this Section, we show that this structure can indeed be preserved, and thus acceleration is possible.

3.1 Accelerating Soft-Impute

To accelerate Soft-Impute, recall from Sections 2.1 and 2.2 that we have to compute $\text{prox}_g(\tilde{Z}_t) = \text{SVT}_\lambda(\tilde{Z}_t)$, where

$$\begin{aligned} \tilde{Z}_t &= Y_t - \nabla f(Y_t) = P_\Omega(O - Y_t) + Y_t \\ &= P_\Omega(O - Y_t) + (1+\theta_t)X_t - \theta_t X_{t-1}. \end{aligned} \tag{10}$$

The efficiency of Soft-Impute hinges on the "sparse plus low-rank" structure of Z_t in (6), which allows matrix-vector multiplications of the form $Z_t u$ and $v^\top Z_t$ to be computed inexpensively. In the following, we show that $\tilde{Z}_t$ also has a similar structure.

Assume that X_t and X_{t-1} have ranks r_t and r_{t-1}, and their SVDs are $U_t \Sigma_t V_t^\top$ and $U_{t-1}\Sigma_{t-1}V_{t-1}^\top$, respectively. Using these SVDs, the sparse $P_\Omega(O - Y_t)$ can be constructed in $O((r_t + r_{t-1})\|\Omega\|_1)$ time. Using (10), for any $u \in \mathbb{R}^n$, $P_\Omega(O - Y_t)u$ can be obtained in $O(\|\Omega\|_1)$ time; and $(1+\theta_t)X_t u - \theta_t X_{t-1} u = (1+\theta_t)U_t\Sigma_t({V_t}^\top u) - \theta_t U_{t-1}\Sigma_{t-1}({V_{t-1}}^\top u)$ in $O((m+n)(r_{t-1}+r_t))$ time. The same applies to the computation of $v^\top \tilde{Z}_t$ for any $v \in \mathbb{R}^m$. Thus, the rank-k SVD of $\tilde{Z}_t$ can be obtained in $O((r_t + r_{t-1} + k)\|\Omega\|_1 + (r_{t-1} + r_t)k(m+n))$ time, which is only about twice that of Soft-Impute.

3.2 Approximating the SVT

Note from (8) that $\text{SVT}_\lambda(\tilde{Z}_t)$ only involves the leading singular vectors of $\tilde{Z}_t$ (whose singular values are greater than or equal to λ), and thus only a partial SVD is needed. Let $\hat{k}$ be the number of such singular values. The power method (Algorithm 1) [Halko *et al.*, 2011] is a fast and accurate algorithm for obtaining an approximation $Q \in \mathbb{R}^{m\times k}$ (where $k \geq \hat{k}$) of such a subspace. Besides the power method, algorithms such as PROPACK [Larsen, 1998] have also been used [Toh and Yun, 2010]. However, the power method is more efficient than PROPACK [Wu and Simon, 2000], and it also allows warm-start, which is particularly useful because of the iterative nature of Soft-Impute.

Algorithm 1 PowerMethod($\tilde{Z}, R, \tilde{\epsilon}$). In step 3, QR($\cdot$) is the QR factorization.

```
Require: $\tilde{Z} \in \mathbb{R}^{m\times n}$, initial $R \in \mathbb{R}^{k\times n}$ for warm-start,
    tolerance $\tilde{\epsilon}$;
1: Initialize $Y_1 \leftarrow \tilde{Z}R$;
2: for $\tau = 1, 2, \dots,$ do
3:     $Q_{\tau+1} = \text{QR}(Y_\tau)$;
4:     $Y_{\tau+1} = \tilde{Z}(\tilde{Z}^\top Q_{\tau+1})$;
5:     if $\|Q_{\tau+1}{Q_{\tau+1}}^\top - Q_\tau {Q_\tau}^\top\|_F \leq \tilde{\epsilon}$ then
6:         break;
7:     end if
8: end for
9: return $Q_{\tau+1}$;
```

With this Q, $\text{SVT}(\tilde{Z}_t)$ can be obtained from a much smaller SVT as follows.

Proposition 3.1. *Let $Q \in \mathbb{R}^{m\times k}$ where $k \geq \hat{k}$, be orthogonal and contains the subspace spanned by the top $\hat{k}$ left singular vectors of $\tilde{Z}_t$. Then, $SVT_\lambda(\tilde{Z}_t) = Q\ SVT_\lambda(Q^\top \tilde{Z}_t)$.*

Algorithm 2 shows how to approximate $\text{SVT}_\lambda(\tilde{Z}_t)$. Let the target (exact) rank-k SVD of $\tilde{Z}_t$ be $\tilde{U}_k \tilde{\Sigma}_k \tilde{V}_k^\top$. Step 1 first approximates $\tilde{U}_k$ by the power method. In steps 2 to 5, a much smaller, and thus less expensive, (exact) $\text{SVT}_\lambda(Q^\top \tilde{Z}_t)$ is obtained from (8). Finally, $\text{SVT}_\lambda(\tilde{Z}_t)$ is recovered using Proposition 3.1.

Algorithm 2 Algorithm to approximate the SVT of $\tilde{Z}_t$ (approx-SVT($\tilde{Z}_t, R, \lambda, \tilde{\epsilon}$)).

```
Require: $\tilde{Z}_t \in \mathbb{R}^{m\times n}$, $R \in \mathbb{R}^{n\times k}$, thresholds $\lambda$ and $\tilde{\epsilon}$;
1: $Q = \text{PowerMethod}(\tilde{Z}_t, R, \tilde{\epsilon})$;
2: $[U, \Sigma, V] = \text{SVD}(Q^\top \tilde{Z}_t)$;
3: $U = \{u_i \mid \sigma_i > \lambda\}$;
4: $V = \{v_i \mid \sigma_i > \lambda\}$;
5: $\Sigma = (\Sigma - \lambda I)_+$;
6: return $QU, \Sigma$ and $V$.
```

The main computation is on the power method (step 1), whose most expensive operations, in turn, are on matrix multiplications (steps 1 and 4 of Algorithm 1). By using the special structure of $\tilde{Z}_t$ in (10) as discussed in Section 3.1, the time complexity of Algorithm 2 can be reduced from $O(mnk)$ to $O((r_t + r_{t-1} + k)\|\Omega\|_1 + (r_{t-1} + r_t)k(m+n))$.

The following Proposition guarantees the quality of $\hat{X}_t = (QU)\Sigma V^\top$ returned from Algorithm 2.

Proposition 3.2. *Let $\sigma_{t,k}$ be the kth singular value of $\tilde{Z}_t$, $\eta_t = \sigma_{t,k+1}/\sigma_{t,k} < 1$, α_t, β_t, γ_t be constants depending on $\tilde{Z}_t$, and that the power method terminates after j iterations. Assume that $k \geq \hat{k}$, and $\tilde{\epsilon} \geq \alpha_t \eta_t^j \sqrt{1+\eta_t^2}$. Then,*

$$h_{\lambda\|\cdot\|_*}(\hat{X}_t; \tilde{Z}_t) \leq h_{\lambda\|\cdot\|_*}(SVT_\lambda(\tilde{Z}_t); \tilde{Z}_t) + \frac{\eta_t}{1-\eta_t}\tilde{\epsilon}\beta_t\gamma_t, \tag{11}$$

where $h_{\lambda\|\cdot\|_*}(X; \tilde{Z}_t) \equiv \frac{1}{2}\|X - \tilde{Z}_t\|^2 + \lambda\|X\|_*$ is the objective in the proximal step.

3.3 Proposed Algorithm

The whole procedure is shown in Algorithm 3. The core steps are 8–11, where an approximate SVT is performed. Similar to [Lin *et al.*, 2010; Toh and Yun, 2010], steps 9–10 use the column space of the last two iterations (V_t and V_{t-1}) to warm-start the power method. For further speedup, at step 5, we employ a continuation strategy as in [Toh and Yun, 2010; Mazumder *et al.*, 2010], in which λ_t is initialized to a large value and then decreased gradually. Moreover, as in [O'Donoghue and Candes, 2012; Nesterov, 2013], the algorithm is restarted (at step 12) if $F(X)$ starts to increase.

3.4 Convergence of the Inexact APG Algorithm

Since $\hat{X}_t$ only approximates $\text{SVT}(\tilde{Z}_t)$, the proximal objective $h_{\lambda\|\cdot\|_*}(\cdot; \tilde{Z}_t)$ is not exactly minimized. The convergence of such inexact APG algorithms has been recently studied in [Schmidt *et al.*, 2011]. In general, for problem (2), an inexact APG algorithm may commit two types of errors: (i) an error e_t in the calculation of $\nabla f(\cdot)$, and (ii) an error ε_t in the proximal objective achieved by x_{t+1}, i.e.,

$$h_{\mu g}(x_{t+1}; z_t) \leq \varepsilon_t + h_{\mu g}(\text{prox}_{\mu g}(z_t); z_t). \tag{12}$$

Algorithm 3 Accelerated Inexact Soft-Impute (AIS-Impute).

```
Require: partially observed matrix $O$, parameter $\lambda$, decay parameter $\nu \in (0,1)$, threshold $\epsilon$;
 1: $[U_0, \lambda_0, V_0] =$ rank-1 SVD$(P_\Omega(O))$;
 2: initialize $c = 1$, $\tilde{\epsilon}_0 = \|P_\Omega(O)\|_F$, $X_0 = X_1 = \lambda_0 U_0 V_0^\top$;
 3: for $t = 1, 2, \dots$ do
 4:     $\tilde{\epsilon}_t = \nu^t \tilde{\epsilon}_0$;
 5:     $\lambda_t = \nu^t(\lambda_0 - \lambda) + \lambda$;
 6:     $\theta_t = (c-1)/(c+2)$;
 7:     $Y_t = X_t + \theta_t(X_t - X_{t-1})$;
 8:     $\tilde{Z}_t = Y_t + P_\Omega(O - Y_t)$;
 9:     $V_{t-1} = V_{t-1} - V_t(V_t^\top V_{t-1})$, remove zero columns;
10:     $R_t = \text{QR}([V_t, V_{t-1}])$;
11:     $[U_{t+1}, \Sigma_{t+1}, V_{t+1}] =$ approx-SVT$(\tilde{Z}_t, R_t, \lambda_t, \tilde{\epsilon}_t)$;
12:     if $F(U_{t+1}\Sigma_{t+1}V_{t+1}^\top) > F(U_t \Sigma_t V_t^\top)$ then
13:         $c = 1$;
14:     else
15:         $c = c + 1$;
16:     end if
17:     if $|F(U_{t+1}\Sigma_{t+1}V_{t+1}^\top) - F(U_t\Sigma_t V_t^\top)| \le \epsilon$ then
18:         break;
19:     end if
20: end for
21: return $X_{t+1} = U_{t+1}\Sigma_{t+1}V_{t+1}^\top$.
```

The following Theorem shows that the convergence rate remains unchanged if the errors decrease at appropriate rates.

Theorem 3.3 ([Schmidt *et al.*, 2011]). *Assume that (i) f is convex and has Lipschitz-continuous gradient; and (ii) g is convex and possibly nonsmooth. If $\|e_t\|$ and $\sqrt{\varepsilon_t}$ decrease as $O(1/t^{2+\delta})$ for some $\delta > 0$, the inexact APG algorithm still converges with rate $O(1/T^2)$.*

In the proposed Algorithm 3, there is no error in $\nabla f(\cdot)$ and so $e_t = 0$. As for ε_t, we have from Proposition 3.2 that $\varepsilon_t \le \frac{\eta_t}{1-\eta_t}\tilde{\epsilon}_t\beta_t\gamma_t$. From step 4, $\tilde{\epsilon}_t = O(\nu^t)$ where $\nu \in (0,1)$. Thus, ε_t converges to zero at a linear rate and is faster than the $O(1/t^{2+\delta})$ rate required in Theorem 3.3. This is formally stated in the following Theorem.

Theorem 3.4. *Assume the conditions in Proposition 3.2, Algorithm 3 converges to the optimal solution of (1) with a rate of $O(1/T^2)$.*

4 Experiments

In this section, we perform experiments on both synthetic data (Section 4.1) and the entire MovieLens and Netflix recommendation data sets (Section 4.2).

4.1 Synthetic Data

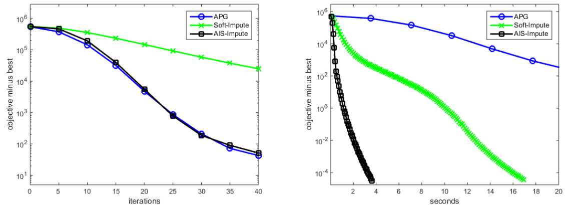
We generate a $m \times m$ data matrix $O = UV + G$, where the elements of $U \in \mathbb{R}^{m\times 5}, V \in \mathbb{R}^{5\times m}$ are sampled i.i.d. from the normal distribution $\mathcal{N}(0,1)$, and elements of G sampled from $\mathcal{N}(0, 0.05)$. A total of $\|\Omega\|_1 = 15m\log(m)$ random elements in O are observed. Half of them are used for training, and the other half as validation set for parameter tuning. Testing is performed on the unobserved (missing) elements.

The following variants of proximal gradient algorithms are compared:

- accelerated proximal gradient algorithm (denoted "APG")[1] [Ji and Ye, 2009; Toh and Yun, 2010];
- Soft-Impute[2] [Mazumder *et al.*, 2010];
- the proposed accelerated inexact Soft-Impute algorithm (denoted "AIS-Impute") in Algorithm 3.

APG and AIS-Impute are in Matlab, while Soft-Impute is in R (and is called from Matlab). For performance evaluation, we use (i) the normalized mean squared error NMSE $= \sqrt{\|P_\Omega^\perp(X - UV)\|_F / \|P_\Omega^\perp(UV)\|_F}$, where X is the recovered matrix; and (ii) the rank of X. We vary m in $\{500, 1000, 1500, 2000\}$. Each experiment is repeated five times. Experiments are performed on a PC with Intel i7 CPU and 16GB RAM.

Results are shown in Table 1. As can be seen, all algorithms are equally good at recovering the missing matrix entries, but AIS-Impute is much faster. A more detailed timing comparison is in Figure 1. APG and AIS-Impute converge much faster than Soft-Impute w.r.t. the number of iterations (Figure 1(a)), as their convergence rates are $O(1/T^2)$ rather than $O(1/T)$. Because of its inexact proximal step, AIS-Impute has a slightly higher objective than APG. However, when measured against time (Figure 1(b)), APG is the slowest as it does not utilize the "sparse plus low-rank" structure. Overall, AIS-Impute is the fastest, as it has both low per-iteration complexity and fast $O(1/T^2)$ convergence rate.



(a) objective vs #iterations. (b) objective vs time.

Figure 1: Performance on synthetic data (with $m = 2000$).

4.2 Recommendation Data

MovieLens Data

The MovieLens data set[3] (Table 2) contains ratings (from 1 to 5) of different users on movies, and has been commonly used in matrix completion experiments [Mazumder *et al.*, 2010; Hsieh and Olsen, 2014]. Following [Wang *et al.*, 2014], we use 50% of the observed ratings for training, 25% for validation and the rest for testing. Besides the proximal gradient algorithms in Section 4.1, we also compare with other state-of-the-art non-proximal-gradient-based matrix completion algorithms, including

[1] http://www.math.nus.edu.sg/~mattohkc/NNLS.html

[2] http://cran.r-project.org/web/packages/softImpute/index.html

[3] http://grouplens.org/datasets/movielens/

Table 1: Results on the synthetic data (time is in seconds). Here, number in brackets is the data sparsity.

	$m = 500$ (18.64%)			$m = 1000$ (10.36%)			$m = 1500$ (7.31%)			$m = 2000$ (5.70%)		
	NMSE	rank	time	NMSE	rank	time	NMSE	rank	time	NMSE	rank	time
APG	0.0183	5	5.1	0.0223	5	45.5	0.0251	5	172.7	0.0273	5	483.9
Soft-Impute	0.0183	5	1.3	0.0223	5	4.4	0.0251	5	13.3	0.0273	5	18.7
AIS-Impute	0.0183	5	0.3	0.0223	5	1.1	0.0251	5	2.0	0.0273	5	2.9

- active subspace selection (denoted "active") [4] algorithm [Hsieh and Olsen, 2014]: In each iteration, this algorithm uses the power method to identify the active row and column subspaces, and then reduces the nuclear norm optimization problem to a smaller problem;
- a Frank-Wolfe based algorithm with local acceleration (denoted "boost")[5] [Zhang *et al.*, 2012];
- a recent variant of Soft-Impute, which replaces the SVD in the soft-thresholding step by alternating least squares (denoted as "ALT-Impute") [Hastie *et al.*, 2014] and
- a second-order trust-region algorithm [6] (denoted "TR") that alternates between fixed-rank optimization and rank-one updates [Mishra *et al.*, 2013].

For performance evaluation, as in [Hsieh and Olsen, 2014; Mazumder *et al.*, 2010], we use (i) the root mean squared error RMSE $= \sqrt{\|P_\Omega(X - O)\|_F^2 / \|\Omega\|_1}$; and (ii) the rank of X. The experiment is repeated five times.

Table 2: Recommendation data sets used in the experiments.

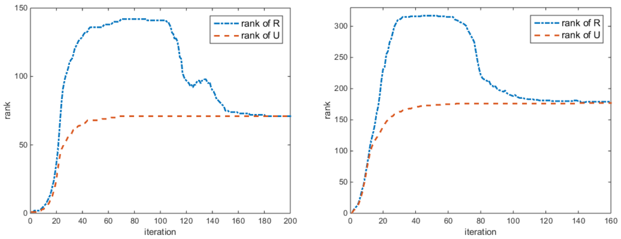
	#users	#movies	#ratings
MovieLens-100K	943	1,682	100,000
MovieLens-1M	6,040	3,449	999,714
MovieLens-10M	69,878	10,677	10,000,054

Results are shown in Table 3, and convergence of the objective and testing RMSE are shown in Figure 2. Again, all algorithms are equally good at recovering the missing matrix entries. In terms of speed, AIS-Impute is the fastest. ALT-Impute has the same convergence rate as Soft-Impute, but is faster (than Soft-Impute) as it does not require performing SVD. As for boost, it only needs to perform a sparse rank-one SVD in each iteration. However, much time is spent on maintaining the recovery matrix in factorized form and also local acceleration. TR is the slowest, as it has to solve many fixed-rank optimization problems.

In Proposition 3.2, we assume that the rank of R_t is not smaller than the rank of U_{t+1} (step 11 of Algorithm 3). Figure 3 compares their values throughout the iterations. As can be seen, this assumption always holds and the two ranks gradually converge to the final recovered rank.

Netflix Data

In this Section, we demonstrate the speedup of AIS-Impute over ALT-Impute and Soft-Impute on the Netflix data set, which contains ratings of 480,189 users on 17,770 movies. 1% of the ratings matrix are observed. We randomly sample 50% of the observed ratings for training, and the rest for testing. As in [Mazumder *et al.* 2010], several choices of λ are used.



(a) MovieLens-100K.

(b) MovieLens-1M.

Figure 3: Ranks of R_t and U_{t+1} as Algorithm 3 iterates.

Results are shown in Table 4, and convergence of the objective and testing RMSE w.r.t. the running time are shown in Figure 4. As can be seen, all algorithms can recover the same RMSE and rank. However, AIS-Impute does not involve with SVD and has a better $O(1/T^2)$ rate. Thus, it is again the fastest.

5 Conclusion

In this paper, we show that Soft-Impute, as a proximal gradient algorithm, can be accelerated without losing the "sparse plus low-rank" structure crucial to the efficiency of Soft-Impute. To further reduce the per-iteration time complexity, we proposed an approximate-SVT scheme based on the power method. Theoretical analysis shows that the proposed algorithm still enjoys the fast $O(1/T^2)$ convergence rate. Extensive experiments on both synthetic and large recommendation data sets show that the proposed algorithm is much faster than the state-of-the-art.

Acknowledgment

This research was supported in part by the Research Grants Council of the Hong Kong Special Administrative Region (Grant No. 614012).

References

[Beck and Teboulle, 2009] A. Beck and M. Teboulle. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM Journal on Imaging Sciences*, 2(1):183–202, 2009.

[Cabral *et al.*, 2011] R.S. Cabral, F. Torre, J.P. Costeira, and A. Bernardino. Matrix completion for multi-label

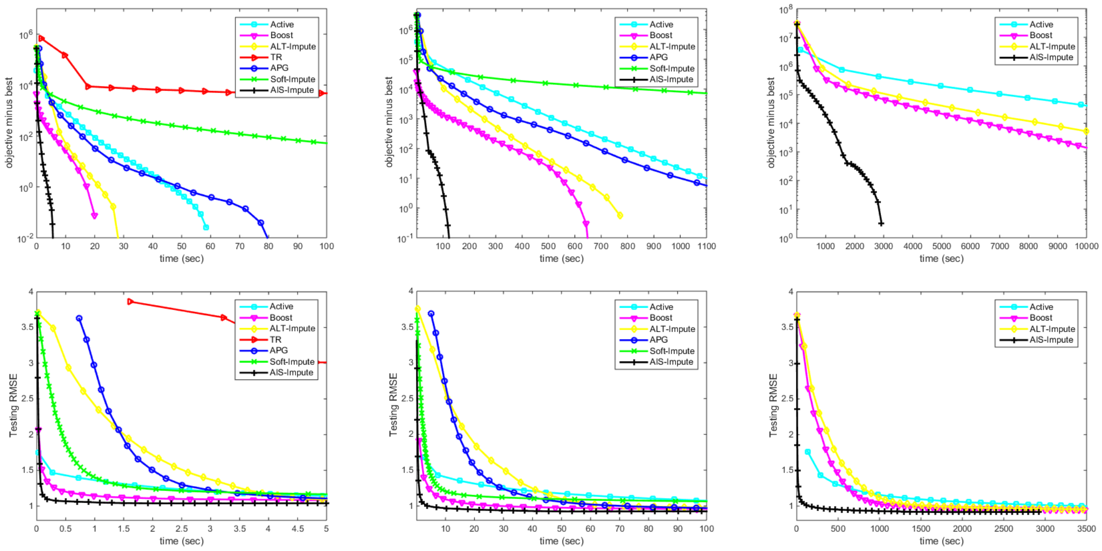
[4] http://www.cs.utexas.edu/~cjhsieh/nuclear_active_1.1.zip

[5] http://users.cecs.anu.edu.au/~xzhang/GCG/

[6] https://sites.google.com/site/bamdevm/codes/tracenorm

Table 3: Results on the MovieLens data sets. Note that many algorithms fail to converge in 10^6 seconds.

	MovieLens-100K			MovieLens-1M			MovieLens-10M		
	RMSE	rank	time	RMSE	rank	time	RMSE	rank	time
active	1.037	70	59.5	0.925	180	1431.4	0.918	217	29681.4
boost	1.038	71	19.5	0.925	178	616.3	0.917	216	13873.9
ALT-Impute	1.037	70	29.1	0.925	179	797.1	0.919	215	17337.3
TR	1.037	71	1911.4	—	—	$> 10^6$	—	—	$> 10^6$
APG	1.037	70	83.4	0.925	180	2060.3	—	—	$> 10^6$
Soft-Impute	1.037	70	337.6	0.925	180	8821.0	—	—	$> 10^6$
AIS-Impute	1.037	70	5.8	0.925	179	129.7	0.916	215	2817.5



(a) MovieLens-100K.

(b) MovieLens-1M.

(c) MovieLens-10M.

Figure 2: Convergence of the objective (top) and testing RMSE (bottom) w.r.t. time on the MovieLens data sets.

image classification. In *Advances in Neural Information Processing Systems*, pages 190–198, 2011.

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

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

[Halko *et al.*, 2011] N. Halko, P.-G. Martinsson, and J.A. Tropp. Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM Review*, 53(2):217–288, 2011.

[Hastie *et al.*, 2014] T. Hastie, R. Mazumder, J. Lee, and R. Zadeh. Matrix completion and low-rank svd via fast alternating least squares. Technical Report arXiv:1410.2596, 2014.

[Hsieh and Olsen, 2014] C.-J. Hsieh and P. Olsen. Nuclear norm minimization via active subspace selection. In *Proceedings of the 31st International Conference on Machine Learning*, pages 575–583, 2014.

[Ji and Ye, 2009] S. Ji and J. Ye. An accelerated gradient method for trace norm minimization. In *Proceedings of the 26th International Conference on Machine Learning*, pages 457–464, 2009.

[Kim and Leskovec, 2011] M. Kim and J. Leskovec. The network completion problem: Inferring missing nodes and edges in networks. In *Proceedings of the 11th SIAM International Conference on Data Mining*, 2011.

[Larsen, 1998] R.M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. DAIMI PB-357, Department of Computer Science, Aarhus University, 1998.

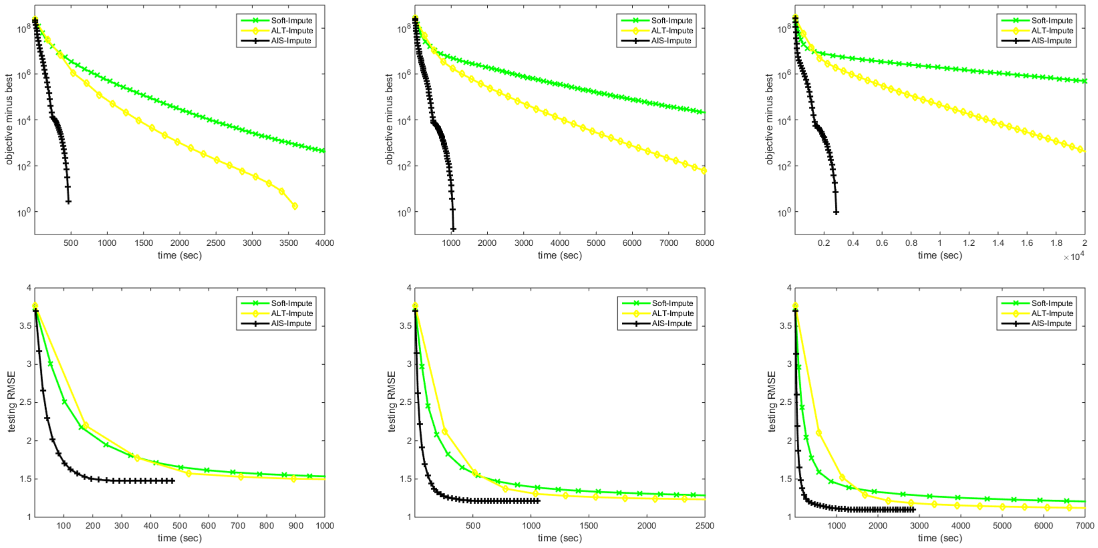
[Lin *et al.*, 2010] Z. Lin, M. Chen, and Y. Ma. The augmented Lagrange multiplier method for exact recovery of corrupted low-rank matrices. Technical Report arXiv:1009.5055, 2010.

[Liu *et al.*, 2013] J. Liu, P. Musialski, P. Wonka, and J. Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[Mazumder *et al.*, 2010] R. Mazumder, T. Hastie, and R. Tibshirani. Spectral regularization algorithms for

Table 4: Results on the Netflix data set. Here, $\lambda_0 = \|P_\Omega(O)\|_F$, and time is in hours.

	$\lambda = \lambda_0/50$			$\lambda = \lambda_0/100$			$\lambda = \lambda_0/150$		
	RMSE	rank	time	RMSE	rank	time	RMSE	rank	time
ALT-Impute	1.480	3	1.02	1.213	5	2.65	1.099	21	9.02
Soft-Impute	1.480	3	1.78	1.213	5	5.75	1.099	21	24.47
AIS-Impute	1.480	3	0.13	1.213	5	0.21	1.099	21	0.79



(a) $\lambda = \lambda_0/50$.

(b) $\lambda = \lambda_0/100$.

(c) $\lambda = \lambda_0/150$.

Figure 4: Convergence of the objective (top) and testing RMSE (bottom) w.r.t. time on the Netflix data set.

learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322, 2010.

[Mishra *et al.*, 2013] B. Mishra, G. Meyer, F. Bach, and R. Sepulchre. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization*, 23(4):2124–2149, 2013.

[Nesterov, 2013] Y. Nesterov. Gradient methods for minimizing composite functions. *Mathematical Programming*, 140(1):125–161, 2013.

[O'Donoghue and Candes, 2012] B. O'Donoghue and E. Candes. Adaptive restart for accelerated gradient schemes. *Foundations of Computational Mathematics*, pages 1–18, 2012.

[Parikh and Boyd, 2014] N. Parikh and S. Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3):127–239, 2014.

[Recht *et al.*, 2010] B. Recht, M. Fazel, and P.A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, 2010.

[Schmidt *et al.*, 2011] M. Schmidt, N.L. Roux, and F.R. Bach. Convergence rates of inexact proximal-gradient methods for convex optimization. In *Advances in Neural Information Processing Systems*, pages 1458–1466, 2011.

[Tibshirani, 2010] R. Tibshirani. Proximal gradient descent and acceleration. Lecture Notes, 2010. http://www.stat.cmu.edu/~ryantibs/convexopt/lectures/08-prox-grad.pdf.

[Toh and Yun, 2010] K.-C. Toh and S. Yun. An accelerated proximal gradient algorithm for nuclear norm regularized linear least squares problems. *Pacific Journal of Optimization*, 6(615-640):15, 2010.

[Wang *et al.*, 2014] Z. Wang, M.-J. Lai, Z. Lu, W. Fan, and J. Davulcu, H.and Ye. Rank-one matrix pursuit for matrix completion. In *Proceedings of the 31st International Conference on Machine Learning*, pages 91–99, 2014.

[Wu and Simon, 2000] K. Wu and H. Simon. Thick-restart Lanczos method for large symmetric eigenvalue problems. *SIAM Journal on Matrix Analysis and Applications*, 22(2):602–616, 2000.

[Zhang *et al.*, 2012] X. Zhang, D. Schuurmans, and Y.-L. Yu. Accelerated training for matrix-norm regularization: A boosting approach. In *Advances in Neural Information Processing Systems*, pages 2906–2914, 2012.