# A two-phase rank-based algorithm for low-rank matrix completion

Tacildo de S. Araújo\*1, Douglas S. Gonçalves<sup>†2</sup>, and Cristiano Torezzan<sup>‡3</sup>

<sup>1</sup>IMECC, University of Campinas, Campinas, Brazil <sup>2</sup>CFM, Federal University of Santa Catarina, Florianópolis, Brazil <sup>3</sup>FCA, University of Campinas, Limeira, Brasil

### Abstract

Matrix completion aims to recover an unknown low-rank matrix from a small subset of its entries. In many applications, the rank of the unknown target matrix is known in advance. In this paper, first we revisit a recently proposed rank-based heuristic for "known-rank" matrix completion and establish a condition under which the generated sequence is quasi-Fejér convergent to the solution set. Then, by including an acceleration mechanism similar to Nesterov's acceleration, we obtain a new heuristic. Even though the convergence of such heuristic cannot be granted in general, it turns out that it can be very useful as a warm-start phase, providing a suitable estimate for the regularization parameter and a good starting-point, to an accelerated Soft-Impute algorithm. Numerical experiments with both synthetic and real data show that the resulting two-phase rank-based algorithm can recover low-rank matrices, with relatively high precision, faster than other well-established matrix completion algorithms.

**Keywords:** Matrix Completion, proximal gradient algorithm, soft-thresholding, recommender systems

## 1 Introduction

The problem of recovering missing entries in a low-rank matrix  $A \in \mathbb{R}^{m \times n}$  can be formulated in terms of a rank minimization problem as

$$\begin{array}{ll}
\text{minimize} & \text{rank}(X) \\
X \in \mathbb{R}^{m \times n} & \text{subject to} & P_{\Omega}(X) = P_{\Omega}(A),
\end{array} \tag{1}$$

where  $\Omega$  denotes the set of indices of the known entries of A and  $P_{\Omega}(\cdot)$  is the projection operator, defined as

$$[P_{\Omega}(X)]_{ij} := \begin{cases} X_{ij}, & \text{if } (i,j) \in \Omega \\ 0, & \text{otherwise,} \end{cases}$$

with  $P_{\Omega}^{\perp}(\cdot)$  defined by  $P_{\Omega}^{\perp}(X) = X - P_{\Omega}(X)$ .

Despite its theoretical importance, problem (1) is non-convex and combinatorially hard for general sets  $\Omega$  [1]. To overcome such disadvantage, several alternatives have been proposed in the literature [2, 3, 4]. A common way to swerve the non-convexity in problem (1) is to replace the rank objective by a convex relaxation such as the nuclear norm  $\|X\|_*$ , as proposed in [5] and [6].

 $<sup>^*</sup>$ e-mail: tacildo.araujo@ifam.edu.br $^\dagger$ e-mail: douglas@mtm.ufsc.br

<sup>&</sup>lt;sup>‡</sup>e-mail: torezzan@unicamp.br

The nuclear norm of a matrix is derived from its Singular Value Decomposition (SVD). Let  $X = U\Sigma V^{\top}$  be the SVD of  $X \in \mathbb{R}^{m\times n}$  and assume that  $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_{\min\{m,n\}} \geq 0$  are its singular values. The nuclear norm of X is defined as  $\|X\|_* := \sum_j \sigma_j$  and it has been used to propose a convex relaxation for problem (1).

The methods studied in this paper rely on a deflated version of the SVD decomposition, calculated by using the so-called Soft-Thresholding (ST) operator, defined as

$$S_{\tau}(M) := U \Sigma_{\tau} V^{\top}, \quad \Sigma_{\tau} = \operatorname{diag}[(\sigma_1 - \tau)_+, \cdots, (\sigma_q - \tau)_+], \tag{2}$$

where  $M = U\Sigma V^{\top}$  is the compact-SVD of a rank q matrix M and  $t_{+} = \max(0, t)$ . It turns out that  $S_{\tau}(M)$  is a proximal operator [7, Theorem 2.1] which solves the problem

$$\min_{X} \frac{1}{2} \|M - X\|_F^2 + \tau \|X\|_*.$$

In [7], based on the Uzawa's method for finding saddle points of the Lagrangian, the authors present an algorithm, called Singular Value Thresholding (SVT), and proved that the sequence generated by

$$X^{k+1} = S_{\tau}(Y^k) \tag{3}$$

$$Y^{k+1} = Y^k + t_k P_{\Omega}(A - X^{k+1}) \tag{4}$$

where  $Y^0 = 0$  and  $t_k$  is a step-size, converges to the unique solution of the following optimization problem

where  $\tau > 0$  is a regularization parameter. The component  $\tau ||X||_*$  in the objective function is a convex relaxation for rank(X), while  $\frac{1}{2}||X||_F^2$  is a strongly convex term granting (5) a unique solution. Due to its theoretical and computational properties, the SVT algorithm is an important reference for matrix completion and it is often used as a benchmark.

Another alternative formulation for problem (1) is to consider a tolerance on the recovering of the known entries. This can be particularly useful in applications where data are obtained through noisy processes. In this case, it may be worth to consider the following optimization problem

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad ||X||_* 
\text{subject to} \quad ||P_O(X - A)||_F^2 < \delta,$$
(6)

where  $\delta \geq 0$  is a given recovering error tolerance.

In [8] and also in [9] it is exploited the following Lagrangian formulation for problem (6),

$$\underset{X \in \mathbb{R}^{m \times n}}{\text{minimize}} \quad \frac{1}{2} \|P_{\Omega}(A) - P_{\Omega}(X)\|_F^2 + \lambda \|X\|_* =: f_{\lambda}(X), \tag{7}$$

where  $\lambda > 0$  is a regularization parameter. The authors in [8] showed that the sequence produced by

$$X^{k+1} = S_{\lambda t_k}(Y^k)$$

$$Y^{k+1} = X^{k+1} + t_k P_{\Omega}(A - X^{k+1})$$
(8)
(9)

$$Y^{k+1} = X^{k+1} + t_k P_{\Omega}(A - X^{k+1}) \tag{9}$$

converges to a solution of (7). This iterative process is called Fixed Point Continuation (FPC). If the step-size is fixed as  $t_k = 1$ , the above iteration reduces to the so-called Soft-Impute (SI) algorithm discussed in [9]. Both, SI and FPC, rely on a pre-specified decreasing sequence of regularization parameters  $\lambda_1 > \cdots > \lambda_K$ , solving (7), up to a predetermined tolerance, for each value of  $\lambda$ .

Although they may have different motivations, the FPC and SI algorithms can be seen as particular cases of the proximal gradient method [10]. This fact was actually used in [11] to derive a convergence analysis for these algorithms and to propose acceleration strategies for the SI.

Such algorithms, however, are still very sensitive to the choice of the regularization parameter  $\lambda$  and the parameter tuning process can be quite cumbersome in real applications. Another information that is disregarded, or not properly used, by these algorithms is the eventual knowledge of the rank of the target matrix. In some applications, such as in problems involving Euclidean Distance Matrices (EDM), the rank of the matrix to be completed is known in advance. For example, it can be proved that the rank of an EDM derived from a set of points in  $\mathbb{R}^d$  is at most d+2 [12]. This information might be useful to estimate the parameter  $\lambda$  and improve the completion performance.

In [13] the authors take into account the rank information and propose an algorithm called *Fixed-Rank Soft-Impute* (FRSI) to complete missing entries in EDMs using the rank information to estimate the regularization parameter  $\lambda$ . However, despite the good numerical results, no convergence analysis was provided for FRSI.

In this paper, we first revisit the rank-based heuristic proposed in [13] and analyze some properties of the operator defining the iterative process. We show that under an assumption on the behavior of the singular values of the iterates, the sequence generated by such heuristic is quasi-Fejér convergent to the set of matrices with rank not greater than the target rank and that agree with the target matrix in the sampled entries. Then, based on acceleration techniques for proximal gradient methods [14], we devise an accelerated heuristic.

Even though the convergence of such heuristic cannot be granted in general, it turns out that it can be very useful as a warm-start phase, to find a suitable estimate for the regularization parameter  $\lambda$  and a good starting-point, to an accelerated Soft-Impute algorithm [11]. The resulting is a two-phase rank-based algorithm for low-rank matrix completion that presents promising results in numerical experiments with both synthetic and real data.

The rest of the paper is organized as follows. Section 2 revisits the algorithm proposed in [13] and proves its convergence, in the quasi-Fejér sense, under an assumption on the behavior of the singular values of the iterates. Since the required assumption is strong, convergence is not granted in general. However, in Section 3, we discuss how such heuristic can be used as a warm-start phase in a two-phase algorithm: the heuristic provides a starting-point and the value for the regularization parameter to be used in an Accelerated Soft-Impute algorithm in the second phase. Section 4 reports some numerical experiments on synthetic and real data and compares the proposed two-phase algorithm with other well-established matrix completion algorithms. Section 5 brings some concluding remarks and discuss directions for future investigations.

## 2 Revisiting fixed-rank Soft-Impute (FRSI) and its convergence

Let  $A \in \mathbb{R}^{m \times n}$  be a matrix with missing entries and rank r, which we assume it is known in advance.

Let us review FRSI [13] by first recalling the iteration of Soft-Impute (SI) [9]. According to the notation used in the Introduction, SI can be described as

$$X^{k} = S_{\lambda} \left( P_{\Omega}(A) + P_{\Omega}^{\perp}(X^{k-1}) \right). \tag{10}$$

This iteration can be deduced by applying the proximal gradient method to the compositive convex optimization problem (7), as discussed in Appendix A.

Notice that for an arbitrary value of  $\lambda$ , there is no reason to expect  $X^k$  to have rank r. However, if we set

$$\lambda = \sigma_{r+1} \left( P_{\Omega}(A) + P_{\Omega}^{\perp}(X^{k-1}) \right),\,$$

i.e., the r+1 largest singular value of  $P_{\Omega}(A) + P_{\Omega}^{\perp}(X^{k-1})$ , then from the definition of  $S_{\lambda}$  in (2) it follows that the rank is at most r for each iterate  $X^k$ . This observation motivated the FRSI algorithm proposed in [13]. However, no convergence analysis was provided in that paper.

By defining  $g(X) = \frac{1}{2} \|P_{\Omega}(X - A)\|_F^2$ , we can write FRSI iteration as:

$$X^{k} = S_{\sigma_{r+1}(X^{k-1} - \nabla g(X^{k-1}))} \left( X^{k-1} - \nabla g(X^{k-1}) \right). \tag{11}$$

Here, we provide some insights on the convergence of iteration (11), by analyzing the operator

$$T(X) := S_{\sigma_{r+1}(X - \nabla g(X))} (X - \nabla g(X)).$$

**Proposition 2.1.** Let  $T: \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$  be the operator defined above and assume that rank(A) = r. Then, the following properties hold.

- (i) T(A) = A
- (ii) If  $B \in \mathbb{R}^{m \times n}$  is such that  $rank(B) \leq r$  and  $P_{\Omega}(B) = P_{\Omega}(A)$ , then T(B) = B
- (iii)  $T(P_{\Omega}^{\perp}(A)) = A$
- (iv)  $T(0) = T(P_{\Omega}(A))$

*Proof.* Observe that  $\nabla g(X) = P_{\Omega}(X - A)$ . (i) Thus, since  $\nabla g(A) = 0$ , and rank(A) = r, it is straightforward that T(A) = A, i.e, A is a fixed-point of T. The same reasoning applies to a matrix B such that  $P_{\Omega}(B) = P_{\Omega}(A)$  and rank $(B) \leq r$ , proofing (ii).

Also notice that

$$P_{\Omega}^{\perp}(A) - \nabla g(P_{\Omega}^{\perp}(A)) = P_{\Omega}^{\perp}(A) - P_{\Omega}\left(P_{\Omega}^{\perp}(A) - A\right) = P_{\Omega}^{\perp}(A) + P_{\Omega}(A) = A,$$

and thus  $T(P_{\Omega}^{\perp}(A)) = A$  as well, showing (iii). Finally, since

$$P_{\Omega}(A) - \nabla q(P_{\Omega}(A)) = P_{\Omega}(A) - P_{\Omega}(P_{\Omega}(A) - A) = P_{\Omega}(A) = 0 - P_{\Omega}(0 - A) = 0 - \nabla q(0),$$

we conclude (iv): 
$$T(0) = T(P_{\Omega}(A))$$
.

Therefore, not only the target matrix A is a fixed-point of T but any other matrix B, of rank at most r, such that  $P_{\Omega}(B) = P_{\Omega}(A)$ . Perhaps, more surprisingly, is the fact that T also admits fixed-points X, such that  $P_{\Omega}(X) \neq P_{\Omega}(A)$ , as show the next proposition.

**Proposition 2.2.** Let  $X \in \mathbb{R}^{m \times n}$  be a matrix of rank at most r, with truncated (r+1)-SVD  $X = U\Sigma V^{\top}$ . If  $\nabla g(X) = -\gamma UV^{\top} - U_{\perp}\Sigma_{\perp}V_{\perp}^{\top}$ , where the columns of  $U_{\perp}$  and  $V_{\perp}$  are orthonormal bases for the orthogonal complement of range of U and V, respectively, and  $\gamma > 0$  with  $\sigma_{\perp}^{\perp} < \gamma$ , for  $i = r+2, \ldots, \min\{m, n\}$ , then X = T(X).

Proof. Observe that

$$X - \nabla g(X) = U(\Sigma + \gamma \mathbf{I}_{r+1})V^{\top} + U_{\perp}\Sigma_{\perp}V_{\perp}^{\top},$$

then, since 
$$\sigma_{r+1}(X - \nabla g(X)) = \gamma > \sigma_i^{\perp}$$
, we obtain  $T(X) = U\Sigma V^{\top} = X$ .

From the above propositions, we see that although the target matrix A is a fixed point of T, which is desirable, in general the operator will not have a unique fixed point and more, there may be fixed points X such that  $P_{\Omega}(X) \neq P_{\Omega}(A)$ . Thus, we cannot expect T to be a contraction.

Nevertheless, we shall see that if the sequence of singular values  $\sigma_{r+1}(X^k - \nabla g(X^k))$  goes to zero fast enough, then we can prove that the sequence  $\{X^k\}$  is quasi-Fejér convergent to the set

$$\mathcal{X}^* = \{ X \in \mathbb{R}^{m \times n} \mid \operatorname{rank}(X) \le r, P_{\Omega}(X) = P_{\Omega}(A) \}.$$

**Definition 2.3.** A sequence  $\{X^k\}$  in  $\mathbb{R}^{m \times n}$  is quasi-Fejér convergent to  $C \subset \mathbb{R}^{m \times n}$  if, for each  $X^* \in C$ , there exists a non-negative summable sequence  $\{\varepsilon_k\}$  such that

$$||X^k - X^*|| \le ||X^{k-1} - X^*|| + \varepsilon_k, \quad k = 1, 2, \dots$$

**Proposition 2.4.** Let  $C \subset \mathbb{R}^{m \times n}$  be a nonempty set and  $\{X^k\}$  a quasi-Fejér sequence convergent to C. Then,

- (i)  $\{X^k\}$  is bounded.
- (ii) If  $\{X^k\}$  has a cluster point  $\bar{X} \in C$ , then the whole sequence  $\{X^k\}$  converges to  $\bar{X}$ .

*Proof.* See [15, Proposition 1]. 
$$\Box$$

**Theorem 2.5.** Let  $\{X^k\}$  be the sequence generated by  $X^k = T(X^{k-1})$ , with  $X^0 \in \mathbb{R}^{m \times n}$ . If the sequence  $\{\sigma_{r+1}(X^k - \nabla g(X^k))\}$  is summable, then  $\{X^k\}$  is quasi-Fejér convergent to the set  $\mathcal{X}^*$ .

*Proof.* Let  $X^* \in \mathcal{X}^*$ . Consider the notation  $\|\cdot\| = \|\cdot\|_F$ ,  $\sigma_{r+1}^k = \sigma_{r+1}(X^k - \nabla g(X^k))$  and  $\sigma_i^{\star} = \sigma_i(X^{\star})$ . Then,

$$||X^{k+1} - X^{\star}|| = ||T(X^{k}) - T(X^{\star})||$$

$$= ||S_{\sigma_{r+1}(X^{k} - \nabla g(X^{k}))}(X^{k} - \nabla g(X^{k})) - S_{\sigma_{r+1}(X^{\star})}(X^{\star})||$$

$$\leq ||S_{\sigma_{r+1}(X^{k} - \nabla g(X^{k}))}(X^{k} - \nabla g(X^{k})) - S_{\sigma_{r+1}(X^{k} - \nabla g(X^{k}))}(X^{\star})||$$

$$+ ||S_{\sigma_{r+1}(X^{k} - \nabla g(X^{k}))}(X^{\star}) - S_{\sigma_{r+1}(X^{\star})}(X^{\star})||$$

$$\leq ||X^{k} - \nabla g(X^{k}) - X^{\star}|| + \left(\sum_{i=1}^{r} ((\sigma_{i}^{\star} - \sigma_{r+1}^{k})_{+} - \sigma_{i}^{\star})^{2}\right)^{1/2}$$

$$\leq ||P_{\Omega}^{\perp}(X^{k} - X^{\star})|| + \sqrt{r}\sigma_{r+1}^{k} \leq ||X^{k} - X^{\star}|| + \sqrt{r}\sigma_{r+1}^{k}$$

$$(12)$$

where we have used the triangle inequality and the nonexpansive property of  $S_{\lambda}(\cdot)$ , with  $\lambda = \sigma_{r+1}(X^k - \nabla g(X^k))$  fixed. Hence, if the sequence  $\{\sigma_{r+1}^k\}$  is summable, then  $\{X^k\}$  is quasi-Fejér convergent to  $\mathcal{X}^*$ .

Therefore, as long as

$$\sum_{k=0}^{\infty} \sigma_{r+1}(X^k - \nabla g(X^k)) < \infty, \tag{13}$$

 $\{X^k\}$  generated by (11) will be quasi-Fejér convergent to  $\mathcal{X}^*$  and, according to Proposition 2.4(ii), if it has a cluster point in this set, the whole sequence will converge to it. Unfortunately, condition (13) is admittedly strong, and does not hold in general. For this reason, FRSI, defined by iteration (11), should be regarded as an heuristic.

#### A two-phase rank-based algorithm 3

Although the iterative process (11) may not converge to a matrix in  $\mathcal{X}^*$ , here we propose to use it, for a fixed number of iterations, as a "warm-start" phase to obtain a good startingpoint and an estimate to the regularization parameter  $\lambda$  (see problem (7)) before applying the Soft-Impute algorithm (see (10)).

This is motivated by our numerical experience with the iterative process (11): we observed that when  $\{X^k\}$  does not converge to an element in  $\mathcal{X}^*$ , it usually converges to an X as in Proposition 2.2 which, although  $P_{\Omega}(\tilde{X}) \neq P_{\Omega}(A)$ , is such that  $\|\tilde{X}\|_{*} < \|A\|_{*}$ , suggesting X as a minimizer of  $\frac{1}{2} \|P_{\Omega}(X) - P_{\Omega}(A)\|_F^2 + \lambda \|X\|_*$  for an appropriate value of  $\lambda > 0$ .

First, inspired by accelerated versions of the proximal gradient method [10], we include an acceleration for FRSI heuristic, resulting in Algorithm 1. This warm-start phase will be called Phase One.

## Algorithm 1 Phase One: Warm-Start

Input: Known entries of  $A \in \mathbb{R}^{m \times n}$  indexed by  $\Omega$ , rank  $r, \epsilon > 0, w \in \mathbb{N}$ , and  $\beta > 0$ .

- 1: Initialize  $X^0 = 0$ ,  $Z^1 = 0$  and  $\rho_0 = \infty$
- 2: **for** j = 1 to w **do**
- Compute the truncated (r+1)-SVD of  $P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^{j})$
- Set  $\rho_j = \sigma_{r+1}(P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^j))$

- if  $|\rho_j \rho_{j-1}|/(1 + \rho_{j-1}) < \epsilon_\rho$  then exit. Compute  $X^j \leftarrow S_{\rho_j} \left( P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^j) \right)$   $Z^{j+1} \leftarrow X^j + \frac{j-1}{j+\beta} \left( X^j X^{j-1} \right)$
- 8: end for

Output:  $Z^j$ ,  $\rho_i$ 

Phase one runs for a pre-specified number w of iterations or until the values of  $\rho_j$  $\sigma_{r+1}(P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^j))$  stabilize. The last value of  $\rho_j$  from phase one is used as regularization parameter  $\lambda$  for the second phase, which consists of an accelerated Soft-Impute algorithm for problem (7), starting from  $Z^{j+1}$ . Phase Two is described in Algorithm 2.

## Algorithm 2 Phase Two: Accelerated Soft-Impute

Input: Known entries of  $A \in \mathbb{R}^{m \times n}$  indexed by  $\Omega$ , rank  $r, \epsilon > 0$ ,  $it_{\text{max}} \in \mathbb{N}, \lambda > 0$  and  $X^0 \in \mathbb{R}^{m \times n}$ 

- 1: Initialize  $Z^1 = X^0$
- 2: for k = 1 to  $it_{\text{max}}$  do
- Compute  $X^k \leftarrow S_{\lambda} \left( P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^k) \right)$
- if some stopping criterion is verified then stop.  $Z^{k+1} \leftarrow X^k + \frac{k-1}{k+2} \left( X^k X^{k-1} \right)$
- 6: end for

Output:  $X^k$ 

**Remark 3.1.** Differently from Phase one, where a truncated (r+1)-SVD was sufficient to evaluate the thresholding operator (because the threshold value was exactly the r+1 largest singular value of  $P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^k)$ , in Phase Two the value of  $\lambda$  is fixed and may be different from  $\sigma_{r+1}(P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^k))$ . As a result, we need to keep an estimate of the rank  $r_k$ , which is updated in each iteration (starting with  $r_1 = r$ ). We compute a truncated  $(r_k + 1)$ -SVD of  $P_{\Omega}(A) + P_{\Omega}^{\perp}(Z^k)$ . If the  $r_k + 1$  singular value is already below the threshold  $\lambda$ , we keep the rank estimate  $r_k$ . Otherwise, we increase  $r_k$  (to  $r_k + 5$ , for example) and repeat the truncated SVD. Finally,  $r_{k+1}$  is set to the number of positive shifted singular values after the last truncated SVD. A similar scheme was used in [7].

## Algorithm 3 Two-phase rank-based algorithm

Input: Known entries of  $A \in \mathbb{R}^{m \times n}$  indexed by  $\Omega$ , rank r,  $\epsilon > 0$ , w,  $it_{\text{max}} \in \mathbb{N}$ , and  $\beta > 0$ .

- 1: Call Algorithm 1 passing  $A, \Omega, r, \epsilon > 0, w$  and  $\beta > 0$
- ▶ Warm-start

- 2: Set  $\lambda = \rho_i$ ,  $X^0 = Z^j$
- 3: Call Algorithm 2 passing  $A,\,\Omega,\,r,\,\epsilon>0,\,it_{\rm max},\,\lambda$  and  $X^0$

 $\triangleright$  Accelerated Soft-Impute

Output:  $X^k$ 

Algorithm 3 summarizes the two-phase rank-based algorithm which uses Algorithm 1 as a warm-start phase (Phase One) and then calls an Accelerated Soft-Impute (Algorithm 2) in the second phase. As we will see in the numerical experiments of Section 4, Algorithm 3 not only outperforms a previous Fixed-Rank Soft-Impute algorithm [13], but is also competitive with well-established algorithms for low-rank matrix completion.

Furthermore, Algorithm 3 has granted convergence to a solution of

$$\min_{X \in \mathbb{R}^{m \times n}} \quad \frac{1}{2} \|P_{\Omega}(X) - P_{\Omega}(A)\|_F^2 + \rho_j \|X\|_* \tag{14}$$

(where  $\rho_i$  is output of Phase One), because Phase One runs for a finite number of iterations and Phase Two is an accelerated proximal gradient method applied to (14) (see Appendix A).

#### Numerical results 4

In this section, we perform matrix completion experiments with both, synthetic data and the MovieLens<sup>2</sup> data set. Moreover, we also provide an empirical study for choosing the acceleration parameter  $\beta$  in Phase One (Algorithm 1).

All the algorithms were implemented in Matlab language and all the numerical results were performed on a PC with Intel Core i7-7500U CPU and 16 GB RAM.

<sup>&</sup>lt;sup>2</sup>A data set which has been widely used in matrix completion experiments and is available in https://grouplens.org/datasets/movielens/.

The proposed Algorithm 3 is compared with those mentioned in Section 1: FRSI, SVT and FPC. All these methods use PROPACK package [16] (more specifically, the routine lansvd which implements a variant of Lanczos algorithm designed for large matrices with sparse plus low-rank structure) for computing only the leading singular values/vectors.

Concerning the stopping criteria for Algorithm 2, we set

$$\min\left\{\frac{|f_{\lambda}(X^k)-f_{\lambda}(X^{k+1})|}{f_{\lambda}(X^k)},\frac{\|X^{k+1}-X^k\|_F}{\|X^k\|_F}\right\}\leq \epsilon_{\lambda},$$

for a given tolerance  $\epsilon_{\lambda} > 0$  and  $f_{\lambda}$  is from (7). For FRSI algorithm we use

$$\min \left\{ \frac{\|P_{\Omega} (X^k - A)\|_F}{\|P_{\Omega} (A)\|_F}, \frac{\|X^{k+1} - X^k\|_F}{\|X^k\|_F} \right\} \le \epsilon_1$$

as the stopping criterion and for SVT and FPC algorithms we follow the recommendations in [7] and [8], and use  $\frac{\|P_{\Omega}(X^k-A)\|_F}{\|P_{\Omega}(A)\|_F} \le \epsilon_2$ , and  $\frac{\|X^{k+1}-X^k\|_F}{\max\{1,\|X^k\|_F\}} \le \epsilon_3$  as the stopping criterion, respectively.

The following procedure were used for generating the synthetic data set: we generated  $n \times n$  matrices of rank  $r \ll n$  of the form  $A = MN \in \mathbb{R}^{n \times n}$ , where the entries of  $M \in \mathbb{R}^{n \times r}$  and  $N \in \mathbb{R}^{r \times n}$  are sampled i.i.d from the standard normal distribution. Then, we deleted, uniformly at random, a percentage p of entries (unobserved entries) of A.

Before we present some numerical results for both synthetic data and MovieLens, we shall give an overview of how to set the parameter  $\beta$  in Algorithm 1.

## 4.1 Tuning the parameter $\beta$

To assess the sensitivity of Algorithm 1 to the parameter  $\beta$ , we performed extensive numerical experiments on the synthetic data set. We set a budget of w=1,000 iterations and vary the problem dimension n, the rank r, percentage of missing data p and the tolerance  $\epsilon_{\rho}$ . The experiments show that the number of iterations of Algorithm 1 (Phase One) can be highly reduced by a suitable choice of the parameter  $\beta$ , mainly when the number of observed entries is very small.

Figure 1 (a) shows the optimal value for  $\beta$  considering the percentage of missing data  $p \in \{92\%, 85\%, 72\%, 50\%\}$ , n = 1000, r = 5, and  $\epsilon_{\rho} = 10^{-8}$ . As can be seen, for  $\beta \ge 19$ , Algorithm 1 reaches the minimum number of iterations in the four scenarios. Furthermore, for p = 92% the number of iterations is reduced by 79% with respect to  $\beta = 2$  (the default value). On the other hand, for  $n \in \{500, 1000, 2000, 4000\}$ , r = 5,  $\epsilon_{\rho} = 10^{-8}$ , and p = 40%, Figure 1 (b) shows the minimum number of iterations in all scenarios for the same value of  $\beta$  ( $\beta \ge 20$ ).

Figure 2 gives us an overview of how to set the value of  $\beta$  as the rank r varies. For this experiment, we fixed  $(n, \epsilon_{\rho}, p) = (1000, 10^{-5}, 50\%)$  and  $r \in \{3, 5, 10, 30, 50, 80, 100\}$ . As can be seen, the bigger is rank of the target matrix the smaller is the "optimal" value of  $\beta$ .

## 4.2 Experiments with synthetic data

Now we turn our attention to experiments with synthetic data, generated as described in the beginning of Section 4. For these experiments, we set n=1000, p=40% and the rank r takes values in the set  $\{10,15,20,40,80,100\}$ . In the stopping criteria, we used the tolerances  $\epsilon_{\rho}=\epsilon_{1}=\epsilon_{2}=10^{-4},\,\epsilon_{3}=10^{-3},\,\epsilon_{\lambda}=10^{-6}$  and for SVT, following [7], we fixed  $\tau=5n$  and  $t_{k}=1.2n^{2}/|\Omega|$ , where  $|\Omega|$  is the cardinality of  $\Omega$ . For FPC we have used the standard strategy to update the regularization parameter:  $\lambda_{0}=\|P_{\Omega}(A)\|_{2},\,\lambda_{k}=\max\{0.25\lambda_{k-1},0.01\}$ , and  $t_{k}=1.99$ , as recommended in [8]. In Algorithm 3, we set the maximum number of iterations of phase one as w=500 as well as the iteration budget for phase two  $it_{\max}=500$ . For the acceleration parameter  $\beta$  we have used  $\{13,13,12,10,5,5\}$ , respectively (following the study of Section 4.1).

For performance evaluation, we use the relative error, defined by  $\text{Rer} = ||A - \tilde{A}||_F / ||A||_F$ , where  $\tilde{A}$  is the recovered matrix and A is the target one. The experimental results are averaged over 5 repetitions.

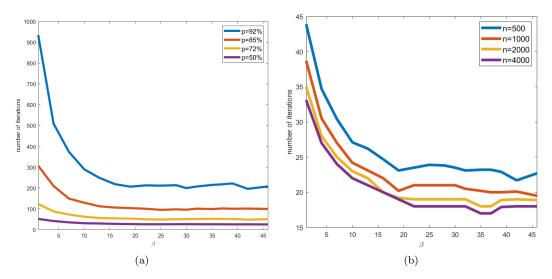


Figure 1: Number of iterations vs  $\beta$ : (a)  $n=1000,\ r=5,\ \epsilon_{\rho}=10^{-8}$  and  $p\in\{92\%,85\%,72\%,50\%\}$ ; (b)  $n\in\{500,1000,2000,4000\},\ r=5,\ \epsilon_{\rho}=10^{-8},\ {\rm and}\ p=40\%.$ 

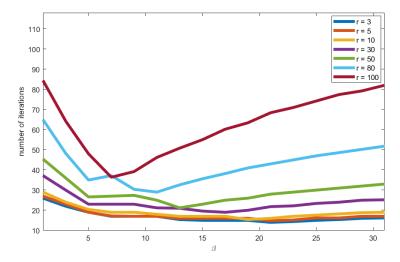


Figure 2: Optimal value for  $\beta$  with  $n=1000, r\in\{3,5,10,30,50,80,100\}, \epsilon_{\rho}=10^{-5}$  and p=50%.

Table 1: Comparison of Algortihm 3, FRSI, SVT, and FPC. Performance evaluation for n = 1,000, p = 40%, r takes values in the set  $\{10,15,20,40,80,100\}$  and  $\beta \in \{13,13,12,10,5,5\}$ , respectively.

$\overline{r}$	method	IT	t(s)	Rer	$\overline{r}$	method	IT	t(s)	Rer
10	Alg. 3	16	1.80	5.84e-06	40	Alg. 3	25	2.77	1.63e-06
	FRSI	18	2.04	1.68e-04		FRSI	28	3.85	2.90e-04
	SVT	43	5.78	1.09e-04		SVT	64	11.15	1.26e-04
	FPC	74	9.68	1.70e-05		FPC	125	56.49	1.83e-05
	Alg. 3	18	1.77	6.90e-06	80	Alg. 3	31	7.08	4.76e-05
1 -	FRSI	20	2.15	1.49e-04		FRSI	42	10.03	5.71e-04
15	SVT	47	5.82	1.07e-04		SVT	93	34.44	1.47e-04
	FPC	81	13.04	1.72e-05		FPC	212	165.98	2.04e-05
20	Alg. 3	18	1.80	1.12e-06	100	Alg. 3	38	12.26	5.42e-05
	FRSI	21	2.89	1.95e-04		FRSI	46	20.51	1.21e-04
	SVT	51	6.60	1.13e-04		SVT	144	68.41	1.76e-04
	FPC	91	15.62	1.78e-05		FPC	361	415.25	2.38e-05

Results are shown in Table 1, where r denotes the target rank, IT is the total number of iterations (for Algorithm 3, it is the sum of iterations of the two phases) and t(s) the time in seconds. In this first set of experiments we point out that all algorithms recovered correctly the underlying rank. As can be seen, our algorithm converges faster than the other algorithms. Furthermore, the bigger is the rank r of the desired matrix the better is the performance of Algorithm 3, when compared with FRSI which, in its turn, is consistently faster than SVT and FPC. In terms of relative error Algorithm 3 was always the first or the second best.

We point out that most of the iterations of Algorithm 3 correspond to phase one (warm-start) iterations. After it switches to the second phase, only a few more iterations are required to reach the stopping criteria. On average, for this set of experiments, the number of phase two iterations is less than 10.

We also performed experiments on larger matrices with very few observed entries. The experiments were conducted under the same parameters as before and we set up a time limit of one hour. We compare the results only with SVT, because, in this case, it is faster than FRSI and FPC algorithms. The results are displayed in Table 2, and it can be seen that both algorithms have competitive performance for the tested cases. In this table we also report an additional column with the recovered rank  $\hat{r}$ .

Algorithm 3 usually outperforms SVT in terms of relative error and it is faster for matrices with higher rank. SVT tends to show a better performance for smaller ranks and when the number of missing entries is not too high. However, it becomes considerably slow when the rank increases and the percentage of known entries decreases. For some cases, such as (1000,20,90%) and (10000,40,97%), we even had to switch to the conservative choice of  $t_k=1.99$ , for which SVT has theoretical convergence guarantees, rather than  $t_k=1.2n^2/|\Omega|$ , to avoid exceed the time limit. Furthermore, we remark that the rank  $\hat{r}$  of the matrix recovered by SVT can be higher than the rank of the original matrix, whereas Algorithm 3 recovered a matrix with correct rank for this set of experiments.

## 4.3 Experiments on *MovieLens* data set

The MovieLens data set is a well-known recommender system that is often used in matrix completion experiments [11]. It contains ratings ( $\{1, 2, 3, 4, 5\}$ ) of different users on movies. Table 3 contains the data sets used in the experiments.

We randomly deleted 50% percent of the observed ratings and for performance evaluation we use the root mean square error (RMSE) given by

$$RMSE = \sqrt{\|P_{\hat{\Omega}}(A - \tilde{A})\|_F^2/|\hat{\Omega}|},$$

where  $\hat{\Omega}$  is total number of observed ratings (but only  $|\Omega| = |\hat{\Omega}|/2$  ratings were passed as input to the algorithms).

Since the ratings matrix has unknown rank and both Algorithm 3 and FRSI need this information, we performed some experiments for different values of r and we set r=130 for MovieLens-100k and r=340 for MovieLens-1M because these choices provide the smallest RMSE for both methods. For Algorithm 3, we fixed the acceleration parameter  $\beta=2$ . In

Table 2: Comparison of Algorithm 3 and SVT for different values of (n, r, p) and  $\beta \in \{13, 12, 19, 12, 19, 12, 19, 10\}$ , respectively.

$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$						
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(n,r,p)	method	IT	t(s)	Rer	$\hat{r}$
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(1000 10 00%)	Alg. 3	116	$5,\!45$	$1.36\mathrm{e}\text{-}04$	10
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(1000,10,9070)	SVT	174	5.26	1.44e-04	10
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(1000.20.00%)	Alg. 3	102	7.34	3.25 e-01	20
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(1000,20,9070)	SVT	500	279.56	$\mathbf{2.23e\text{-}01}$	168
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(2000 10 00%)	Alg. 3	86	12.54	3.68e-05	10
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(2000,10,9070)	SVT	83	$\boldsymbol{9.55}$	1.39e-04	10
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	(2000 20 02%)	Alg. 3	147	28.11	1.59e-04	20
(5000,10,90%)         SVT         53         33.7         1.18e-04         10           (5000,25,96%)         Alg. 3 SVT         215         149.89         1.62e-04         25           (10000,10,90%)         Alg. 3 SVT         65         245.13         8.27e-06         10           (10000,40,97%)         Alg. 3 Alg. 3         256         1018.56         8.01e-04         40	(2000,20,9270)	SVT	262	168.62	1.51e-04	31
SVT         53         33.7         1.18e-04         10           (5000,25,96%)         Alg. 3 SVT         215         149.89         1.62e-04         25           (10000,10,90%)         Alg. 3 SVT         65         245.13         8.27e-06         10           (10000,40,97%)         Alg. 3 Alg. 3         256         1018.56         8.01e-04         40	(5000 10 00%)	Alg. 3	69	63.4	2.36 e-05	10
(5000,25,96%)         SVT         297         1355.23         2.34e-04         50           (10000,10,90%)         Alg. 3 SVT         65 43         245.13 113.84         8.27e-06 10         10           (10000,40,97%)         Alg. 3 Alg. 3         256         1018.56         8.01e-04         40	(5000,10,9070)	SVT	53	33.7	1.18e-04	10
SV I         297         1355.23         2.34e-04         50           (10000,10,90%)         Alg. 3         65         245.13         8.27e-06         10           SVT         43         113.84         1.07e-04         10           (10000,40,97%)         Alg. 3         256         1018.56         8.01e-04         40	(5000.25.06%)	Alg. 3	215	149.89	1.62e-04	25
(10000,10,90%) SVT <b>43</b> 113.84 1.07e-04 10 (10000,40,97%) Alg. 3 <b>256</b> 1018.56 <b>8.01e-04</b> 40	(5000,25,9070)	SVT	297	1355.23	2.34e-04	50
(10000 40 97%) Alg. 3 256 1018.56 8.01e-04 40	(10000 10 00%)	Alg. 3	65	245.13	8.27e-06	10
(100004097%)	(10000,10,9070)	SVT	43	113.84	1.07e-04	10
(10000, 40, 97.70)   SVT   677 3600 4.13e-02 95	(10000 40 07%)	Alg. 3	256	1018.56	8.01e-04	40
	(10000,40,97%)	SVT	677	3600	4.13e-02	95

Table 3: MovieLens data sets used in the experiments

data set	# users	# movies	# ratings
MovieLens-100k	943	1,682	100,000
MovieLens-1M	6,040	3,952	1,000,209

these experiments, we set the tolerances  $\epsilon_{\rho} = \epsilon_1 = \epsilon_2 = \epsilon_3 = 10^{-3}$ ,  $\epsilon_{\lambda} = 10^{-2}$ , and for SVT we fixed  $t_k = 1.99$  and since  $m \neq n$  we set  $\tau = 8\sqrt{mn}$  as suggested in [7]. Moreover, we set up a time limit of one hour for all the algorithms.

The results are shown in Table 4. As we can see, Algorithm 3 shows the best performance in terms of CPU time and RMSE. We remark that for the dataset MovieLens-1M, Algorithm 3 was the only one able to reach the stopping criteria in less than one hour.

## 5 Conclusion

We consider matrix completion problems where the rank of the target matrix is known in advance. For instance, this is the case of localization, graph realization and other problems in distance geometry [17] where the rank of the matrix to be completed is related to the embedding dimension.

We revisited a Fixed Rank Soft-Impute (FRSI) heuristic and, by analyzing the operator defining its iteration, we shown that the generated sequence is quasi-Fejér convergent to  $\mathcal{X}^*$ , under a strong assumption on the behavior of the underlying singular values. Nevertheless, regardless of this assumption, an accelerated version of FRSI can still be helpfull as a warm-start phase for an accelerated Soft-Impute algorithm aimed to solve a nuclear norm regularized least-squares problem. This idea gave rise to a two-phase rank-based algorithm (Algorithm 3) which takes into account the rank information in the heuristic of the first phase to estimate the nuclear norm regularization parameter and provide a warm starting-point to

Table 4: Numerical results on *MovieLens* data sets

method		100k		1M			
method	IT	t(s)	RMSE	IT	t(s)	RMSE	
Alg. 3	84	61.43	0.7667	74	967.76	0.7123	
FRSI	223	159.62	0.8598	176	3,600	0.8475	
SVT	2,000	1,315.92	0.7696	1236	3,600	0.7230	
FPC	410	978.20	0.7806	234	3,600	0.7929	

an accelerated Soft-Impute algorithm at the second phase.

After a numerical study on how to tuning parameters of the first phase, numerical experiments on both synthetic and real data sets indicates that the proposed algorithm (Alg. 3) outperforms the previous heuristic FRSI [13] and is competitive with well established algorithms for matrix completion, such as SVT and FPC. Moreover, Algorithm 3 was able to recover low-rank matrices from a few percentage of its entries with reasonable accuracy and faster than the compared methods, mainly when the expected rank is not too low.

Even though Algorithm 1 (phase-one) is just an heuristic, it was responsible for the majority of the iterations of Algorithm 3. This fact points in the direction of studying convergence properties of phase-one alone under weaker assumptions yet to be discovered.

## References

- [1] N. Srebro and T. Jaakkola, "Weighted low-rank approximations," in *Proceedings of the* 20th International Conference on Machine Learning (ICML-03), pp. 720-727, 2003.
- [2] E. J. Candès and T. Tao, "The power of convex relaxation: Near-optimal matrix completion," *IEEE Transactions on Information Theory*, vol. 56, no. 5, pp. 2053–2080, 2010.
- [3] M. Fornasier, H. Rauhut, and R. Ward, "Low-rank matrix recovery via iteratively reweighted least squares minimization," SIAM Journal on Optimization, vol. 21, no. 4, pp. 1614–1640, 2011.
- [4] J. Tanner and K. Wei, "Low rank matrix completion by alternating steepest descent methods," Applied and Computational Harmonic Analysis, vol. 40, no. 2, pp. 417–429, 2016.
- [5] E. J. Candès and B. Recht, "Exact matrix completion via convex optimization," Foundations of Computational mathematics, vol. 9, no. 6, pp. 717–772, 2009.
- [6] M. Fazel, Matrix rank minimization with applications. PhD thesis, PhD thesis, Stanford University, 2002.
- [7] J.-F. Cai, E. J. Candès, and Z. Shen, "A singular value thresholding algorithm for matrix completion," SIAM Journal on Optimization, vol. 20, no. 4, pp. 1956–1982, 2010.
- [8] S. Ma, D. Goldfarb, and L. Chen, "Fixed point and Bregman iterative methods for matrix rank minimization," *Mathematical Programming*, vol. 128, no. 1, pp. 321–353, 2011.
- [9] R. Mazumder, T. Hastie, and R. Tibshirani, "Spectral regularization algorithms for learning large incomplete matrices," *Journal of Machine Learning Research*, vol. 11, pp. 2287–2322, 2010.
- [10] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," SIAM Journal on Imaging Sciences, vol. 2, no. 1, pp. 183–202, 2009.
- [11] Q. Yao and J. T. Kwok, "Accelerated and inexact soft-impute for large-scale matrix and tensor completion," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 9, pp. 1665–1679, 2018.
- [12] I. Dokmanic, R. Parhizkar, J. Ranieri, and M. Vetterli, "Euclidean distance matrices: essential theory, algorithms, and applications," *IEEE Signal Processing Magazine*, vol. 32, no. 6, pp. 12–30, 2015.
- [13] N. J. Moreira, L. T. Duarte, C. Lavor, and C. Torezzan, "A novel low-rank matrix completion approach to estimate missing entries in euclidean distance matrix," *Computational and Applied Mathematics*, vol. 37, no. 4, pp. 4989–4999, 2018.
- [14] N. Parikh and S. Boyd, "Proximal algorithms," Foundations and Trends in optimization, vol. 1, no. 3, pp. 127–239, 2014.

- [15] A. N. Iusem, "On the convergence properties of the projected gradient method for convex optimization," Computational and Applied Mathematics, vol. 22, no. 1, pp. 37–52, 2003.
- [16] R. M. Larsen, "Lanczos bidiagonalization with partial reorthogonalization," DAIMI Report Series, no. 537, 1998.
- [17] L. Liberti, C. Lavor, N. Maculan, and A. Mucherino, "Euclidean distance geometry and applications," *SIAM review*, vol. 56, no. 1, pp. 3–69, 2014.

## A Soft-Impute as a proximal gradient method

Here we show that Soft-Impute is a particular case of proximal gradient applied to problem (7) with constant step-size. The optimization problem given by equation (7) is a particular case of the problem of minimizing composite functions of the form

$$\underset{x}{\text{minimize}} \quad g(x) + h(x), \tag{15}$$

where g, h are convex functions with g differentiable, having Lipschitz gradient  $\nabla g$  with constant L > 0 (h does not need to be smooth, only proper convex).

Problem (15) can be solved by the proximal gradient algorithm, which generates a sequence  $\{x^k\}$  given by

$$x^{k+1} = \operatorname{prox}_{th} \left( x^k - t \nabla g(x^k) \right), \tag{16}$$

where t > 0 and  $\operatorname{prox}_{th}(\cdot)$  is the proximal operator, which can be expressed as

$$\operatorname{prox}_{th}(v) = \underset{x}{\operatorname{arg\,min}} \left\{ \frac{1}{2t} ||x - v||_{2}^{2} + h(x) \right\}.$$

It is shown (see Theorem 3.1 in [10]) that either for a fixed stepsize  $t \leq \frac{1}{L}$  or by a backtracking line search, the proximal algorithm converges to the optimal solution of (15) at a rate of O(1/k), where k is the number of iterations.

For the function  $h(X) = \lambda ||X||_*$ , the proximal operator is defined as

$$\operatorname{prox}_{th}(M) = \operatorname*{arg\,min}_{X} \left\{ \frac{1}{2t} \| M - X \|_F^2 + \lambda \| X \|_* \right\},$$

whose solution is given by (see Theorem 2.1 in [7] with  $\tau = \lambda t$ )

$$\operatorname{prox}_{th}(M) = S_{\lambda t}(M). \tag{17}$$

In order to show that SI is a proximal gradient algorithm applied to problem (7), let us re-write iteration (10) equivalently as

$$Y^k = X^k + P_{\Omega}(A - X^k) = P_{\Omega}(A) + P_{\Omega}^{\perp}(X^k)$$
  
$$X^{k+1} = S_{\lambda}(Y^k).$$

For problem (7), observe that  $g(X) = \frac{1}{2} \|P_{\Omega}(A) - P_{\Omega}(X)\|_F^2$ , and thus L = 1. Since

$$X^{k} - \nabla g(X^{k}) = X^{k} - (P_{\Omega}(X^{k}) - P_{\Omega}(A)) = P_{\Omega}(A) + P_{\Omega}^{\perp}(X^{k}) =: Y^{k},$$

from (16) and (17) with t = 1 we have  $X^{k+1} = \operatorname{prox}_h(Y^k) = S_{\lambda}(Y^k)$ , which gives the result. One can accelerate the proximal gradient method to achieve the optimal convergence rate of  $O(1/k^2)$  by setting the equations [10, 14]

$$x^{k+1} = \operatorname{prox}_{th} \left( z^k - t \nabla g(z^k) \right)$$
$$z^{k+1} = x^{k+1} + \frac{k-1}{k+2} \left( x^{k+1} - x^k \right)$$
(18)

Therefore, in the same way Soft-Impute corresponds to a proximal gradient method with fixed step-size t = 1, Algorithm 2 corresponds to an accelerated proximal gradient, for which the convergence is well-studied in the literature [14].