




## Intelligent Initialization and Adaptive Thresholding for Iterative Matrix Completion; Some Statistical and Algorithmic Theory for *Adaptive-Impute*

Juhee Cho, Donggyu Kim & Karl Rohe


To cite this article: Juhee Cho, Donggyu Kim & Karl Rohe (2018): Intelligent Initialization and Adaptive Thresholding for Iterative Matrix Completion; Some Statistical and Algorithmic Theory for *Adaptive-Impute*, Journal of Computational and Graphical Statistics, DOI: [10.1080/10618600.2018.1518238](https://doi.org/10.1080/10618600.2018.1518238)

To link to this article: <https://doi.org/10.1080/10618600.2018.1518238>

 View supplementary material 

 Accepted author version posted online: 05 Sep 2018.

 Submit your article to this journal 

 Article views: 78

 View Crossmark data 

# Intelligent Initialization and Adaptive Thresholding for Iterative Matrix Completion; Some Statistical and Algorithmic Theory for *Adaptive-Impute*

Juhee Cho, Donggyu Kim, and Karl Rohe\*

Department of Statistics, University of Wisconsin-Madison

August 27, 2018

## Abstract

Over the past decade, various matrix completion algorithms have been developed. Thresholded singular value decomposition (SVD) is a popular technique in implementing many of them. A sizable number of studies have shown its theoretical and empirical excellence, but choosing the right threshold level still remains as a key empirical difficulty. This paper proposes a novel matrix completion algorithm which iterates thresholded SVD with theoretically-justified and data-dependent values of thresholding parameters. The estimate of the proposed algorithm enjoys the

---

\*This research is supported by NSF grant DMS-1309998 and ARO grant W911NF-15-1-0423.

minimax error rate and shows outstanding empirical performances. The thresholding scheme that we use can be viewed as a solution to a non-convex optimization problem, understanding of whose theoretical convergence guarantee is known to be limited. We investigate this problem by introducing a simpler algorithm, generalized-*softImpute*, analyzing its convergence behavior, and connecting it to the proposed algorithm.

*Keywords:* *softImpute*, generalized-*softImpute*, non-convex optimization, thresholded singular value decomposition

## 1 Introduction

Matrix completion appears in a variety of areas where it recovers a low-rank matrix from a small fraction of observed entries, ranging from collaborating filtering (the ‘Netflix’ problem) to the global positioning problem. Over the past decade, various matrix completion algorithms have been developed (e.g. Rennie and Srebro (2005), Cai et al. (2010), Krishnan et al. (2009), Mazumder et al. (2010), Hastie et al. (2014)). Many of these algorithms employ as a key technique the thresholded singular value decomposition (SVD). The statistical literature has responded by investigating its theoretical optimality (Candès and Plan (2010), Negahban and Wainwright (2011), Koltchinskii et al. (2011)) and strong empirical performances (Rennie and Srebro (2005), Hastie et al. (2014)). However, a key empirical difficulty of employing thresholded SVD for matrix completion is to find the right way and level of threshold (e.g. tuning thresholding parameters, including a rank constraint or not, etc.). Depending on the choice of the thresholding scheme, the rank of the estimated low-rank matrix and predicted values for unobserved entries can widely change. So, there have been some studies (Gavish and Donoho (2014), Donoho and Gavish (2014)) on this issue.

We propose a novel iterative matrix completion algorithm, *Adaptive-Impute*, which recovers the underlying low-rank matrix from a few noisy entries via differentially and adaptively thresholded SVD. Specifically, the proposed *Adaptive-Impute* algorithm differentially thresholds the singular values and adaptively updates the threshold levels on every iteration. As was the case with adaptive Lasso (Zou (2006)) and adaptive thresholding for sparse covariance matrix estimation (Cai and Liu (2011)), the proposed adaptive thresholding scheme gives *Adaptive-Impute* stronger empirical performances than *softImpute* (Mazumder et al. (2010)) and other techniques which use a single thresholding parameter for all singular values throughout the iterations. *Adaptive-Impute* uses a data-dependent and theoretically-justified technique to specify these multiple thresholding parameters that are changing over iterations. Hence, *Adaptive-Impute* is free of the tuning problems regarding the choice of threshold levels. Its single tuning parameter is the rank of the resulting estimator. To choose this tuning parameter, Section 5.2 suggests looking for a gap in the scree plot of singular values of the observed data matrix. Lemma 2 in Cho et al. (2016)

shows that this is a consistent estimator for the true rank.

*Adaptive-Impute* was motivated by the fact that when a fully-observed noisy low-rank matrix is available, the thresholding scheme of *Adaptive-Impute* provides an intuitively-straightforward and theoretically-justified estimator of the underlying true low-rank matrix. In the setting we study in this paper, this helps understand the role of thresholds and rank constraints and what bias or error they can eliminate.

The novel threshold scheme of *Adaptive-Impute* is attempting to solve a non-convex optimization problem. As such, the theoretical guarantees of this algorithm are challenging. To understand the convergence behavior of *Adaptive-Impute*, we introduce a simpler algorithm, *generalized-softImpute* which is also attempting to solve a non-convex optimization problem, and derive sufficient conditions under which it converges. Then, we prove that *Adaptive-Impute* behaves almost the same as *generalized-softImpute* and satisfies its sufficient conditions for convergence. *Generalized-softImpute* helps theoretical understanding on the convergence behavior of itself and *Adaptive-Impute*, but we do not suggest using *generalized-softImpute* in practice since it has lots of parameters that need to be tuned. Numerical experiments and a data analysis in Section 5 suggest superior performances of *Adaptive-Impute* over the existing *softImpute*-type algorithms.

The rest of this paper is organized as follows. Section 2 describes the model setup. Section 3 introduces the proposed algorithm *Adaptive-Impute*. Section 4 introduces a *generalized-softImpute*, a simpler algorithm than *Adaptive-Impute*. Section 5 presents numerical experiment results. Section 6 concludes the paper with discussion. All proofs are collected in the Supplement.

## 2 The model setup

Suppose that we have an  $n \times d$  matrix of rank  $r$ ,

$$M_0 = U\Lambda V^T, \tag{1}$$

where by SVD,  $U = (U_1, \dots, U_r) \in \mathbb{R}^{n \times r}$ ,  $V = (V_1, \dots, V_r) \in \mathbb{R}^{d \times r}$ ,  $\Lambda = \text{diag}(\lambda_1, \dots, \lambda_r) \in \mathbb{R}^{r \times r}$ , and  $\lambda_1 \geq \dots \geq \lambda_r \geq 0$ . The entries of  $M_0$  are corrupted by noise  $\epsilon \in \mathbb{R}^{n \times d}$  whose entries are i.i.d. sub-Gaussian random variables with mean zero and variance  $\sigma^2$ . Hence,

we can only observe  $M_F = M_0 + \epsilon$ . However, oftentimes in real world applications, not all entries of  $M_F$  are observable. So, define  $y \in \mathbb{R}^{n \times d}$  such that  $y_{ij} = 1$  if the  $(i, j)$ -th entry of  $M_F$  is observed and  $y_{ij} = 0$  if it is not observed. The entries of  $y$  are assumed to be i.i.d. Bernoulli( $p$ ) and independent of the entries of  $\epsilon$ . Then, the partially-observed noisy low-rank matrix  $M \in \mathbb{R}^{n \times d}$  is written as

$$M_{ij} = y_{ij}M_{Fij} = \begin{cases} M_{0ij} + \epsilon_{ij} & \text{if observed } (y_{ij} = 1) \\ 0 & \text{otherwise } (y_{ij} = 0). \end{cases}$$

Throughout the paper, we assume that  $r \ll d \leq n$  and the entries of  $M_0$  are bounded by a positive constant  $L$  in absolute value. In this paper, we develop an iterative algorithm to recover  $M_0$  from  $M$  and investigate its theoretical properties and empirical performances.

## 3 Adaptive-Impute algorithm

### 3.1 Initialization

We first introduce some notation. Let a set  $\Omega$  contain indices of the observed entries,  $y_{ij} = 1 \Leftrightarrow (i, j) \in \Omega$ . Then, for any matrix  $A \in \mathbb{R}^{n \times d}$ , denote by  $\mathcal{P}_\Omega(A)$  the projection of  $A$  onto  $\Omega$  and by  $\mathcal{P}_\Omega^\perp(A)$  the projection of  $A$  onto the complement of  $\Omega$ ;

$$[\mathcal{P}_\Omega(A)]_{ij} = \begin{cases} A_{ij} & \text{if } (i, j) \in \Omega \\ 0 & \text{if } (i, j) \notin \Omega \end{cases} \quad \text{and} \quad [\mathcal{P}_\Omega^\perp(A)]_{ij} = \begin{cases} 0 & \text{if } (i, j) \in \Omega \\ A_{ij} & \text{if } (i, j) \notin \Omega. \end{cases}$$

That is,  $\mathcal{P}_\Omega(A) + \mathcal{P}_\Omega^\perp(A) = A$ . We let  $\lambda_i(A)$  the  $i$ -th singular value of  $A$  such that  $\lambda_1(A) \geq \dots \geq \lambda_d(A)$ ,  $\mathbf{u}_i(A)$  denote the  $i$ -th left singular vector of  $A$ , and  $\mathbf{v}_i(A)$  the  $i$ -th right singular vector of  $A$ . Note that in cases where  $\lambda_i(A) = \lambda_{i+1}(A)$ , the corresponding left and right singular vectors span some subspace. In such cases we let any set of orthonormal vectors spanning this subspace can be used for the corresponding left and right singular vectors  $\{\mathbf{u}_i(A), \mathbf{u}_{i+1}(A)\}$  and  $\{\mathbf{v}_i(A), \mathbf{v}_{i+1}(A)\}$  (see Horn and Johnson (1990) for more details). The squared Frobenius norm is defined by  $\|A\|_F^2 = \text{tr}(A^T A)$ , the trace of  $A^T A$ , and the nuclear norm by  $\|A\|_* = \sum_{i=1}^d \lambda_i(A)$ , the sum of the singular values of  $A$ . For a symmetric matrix  $A \in \mathbb{R}^{n \times n}$ ,  $\text{diag}(A)$  represents a matrix with diagonal elements of  $A$  on the diagonal and zeros elsewhere.

Many of the iterative matrix completion algorithms (e.g. Cai et al. (2010), Mazumder et al. (2010), Keshavan et al. (2009), Chatterjee (2014)) in the current literature initialize with  $M$ , where the unobserved entries begin at zero. This initialization works well with algorithms that are based on convex optimization or that are robust to the initial value. However, for algorithms that are based on non-convex optimization or that are sensitive to the initial value, filling the unobserved entries with zeros may not be a good choice. Cho et al. (2016) proposed a one-step consistent estimator,  $\hat{M}$ , that attains the minimax error rate (Koltchinskii et al. (2011)),  $r/pd$ , and requires only two eigendecompositions. *Adaptive-Impute* employs the entries of this one-step consistent estimator instead of zeros as initial values of the unobserved entries. Algorithm 1 describes how to compute the initial value  $\hat{M}$  of *Adaptive-Impute*. The following theorem shows that  $\hat{M}$  achieves the minimax error rate.

---

**Algorithm 1** Initialization (Cho et al. (2016))

---

**Require:**  $M$ ,  $y$ , and  $r$

$$\begin{aligned} \hat{p} &\leftarrow \frac{1}{nd} \sum_{i=1}^n \sum_{j=1}^d y_{ij} \\ \Sigma_{\hat{p}} &\leftarrow M^T M - (1 - \hat{p}) \text{diag}(M^T M) \\ \Sigma_{t\hat{p}} &\leftarrow M M^T - (1 - \hat{p}) \text{diag}(M M^T) \\ \hat{V}_i &\leftarrow \mathbf{v}_i(\Sigma_{\hat{p}}), \quad \forall i \in \{1, \dots, r\} \\ \hat{U}_i &\leftarrow \mathbf{u}_i(\Sigma_{t\hat{p}}), \quad \forall i \in \{1, \dots, r\} \\ \tilde{\alpha} &\leftarrow \frac{1}{d-r} \sum_{i=r+1}^d \lambda_i(\Sigma_{\hat{p}}) \\ \hat{\tau}_i &\leftarrow \lambda_i(\Sigma_{\hat{p}}) - \frac{1}{\hat{p}} \sqrt{\lambda_i(\Sigma_{\hat{p}}) - \tilde{\alpha}}, \quad \forall i \in \{1, \dots, r\} \\ \hat{\lambda}_i &\leftarrow \lambda_i(\Sigma_{\hat{p}}) - \hat{\tau}_i, \quad \forall i \in \{1, \dots, r\} \\ \hat{s}_i &\leftarrow \text{sign}(\langle \hat{V}_i, \mathbf{v}_i(M) \rangle) \text{sign}(\langle \hat{U}_i, \mathbf{u}_i(M) \rangle), \quad \forall i \in \{1, \dots, r\} \\ \hat{M} &\leftarrow \sum_{i=1}^r \hat{s}_i \hat{\lambda}_i \hat{U}_i \hat{V}_i^T \\ \textbf{return } &\hat{M} \end{aligned}$$


---

**Assumption 1.**

- (1)  $pd/\log n \rightarrow \infty$  and  $n, d \rightarrow \infty$  with  $d \leq n \leq e^{d^\beta}$ , where  $\beta < 1$  free of  $n$ ,  $d$ , and  $p$ ;
- (2)  $\lambda_i = b_i \sqrt{nd}$  for all  $i = 1, \dots, r$ , where  $\{b_i\}_{i=1, \dots, r}$  are positive bounded values;

(3)  $b_i > b_{i+1}$  for all  $i = 1, \dots, r$ , where  $b_{r+1} = 0$ ;

**Remark 1.** Under the setting where the rank  $r$  is fixed as in this paper, Assumption 1(2) implies that the underlying low-rank matrix  $M_0$  is dense. More specifically, note that the squared Frobenius norm indicates both the sum of all squared entries of a matrix and the sum of its singular values squared. Also, note that  $\|M_0\|_F^2 = \sum_{i=1}^r \lambda_i^2(M_0) = cnd$  for some constant  $c > 0$  by Assumption 1(2). Thus, the sum of all squared entries of  $M_0$  has an order  $nd$ . This means that a non-vanishing proportion of entries of  $M_0$  contains non-vanishing signals with dimensionality (see Fan et al. (2013)).

**Remark 2.** Assumption 1(2) means that we are assuming “strong” factors, while there are other works studying the “weak” factor regime of  $\lambda_i = b_i\sqrt{n}$ . The types of applications we mainly consider is collaboration filtering, where the entries of  $M_0$  usually represent movie rating from the users, customer reviews, etc.. The entries may not be able to be observed, but it is hard to imagine that they are zeros or sparse. Hence, we believed that Assumption 1(2) serves its purpose.

**Remark 3.** The singular vectors,  $\{\hat{U}_i\}_{i=1}^r$  and  $\{\hat{V}_i\}_{i=1}^r$ , that compose  $\hat{M}$  are consistent estimators of  $U$  and  $V$  up to signs (for details, see Cho et al. (2016)). Hence, when combining them with  $\{\hat{\lambda}_i\}_{i=1}^r$  to reconstruct  $\hat{M}$ , a sign problem happens. Assumption 1(4) assures that as  $n$  and  $d$  increase, the probability of choosing different signs than the true signs,  $\{s_{0i}\}_{i=1}^r$ , goes to zero. Given the asymptotic consistency of  $\{\hat{U}_i\}_{i=1}^r$ ,  $\{\hat{V}_i\}_{i=1}^r$ , and  $\{\hat{\lambda}_i\}_{i=1}^r$ , this is not an unreasonable assumption to make.

**Proposition 3.1.** Under Assumption 1 and the model setup in Section 2, we have

$$\lim_{n,d \rightarrow \infty} \mathbb{P}\left(\left\|\mathcal{P}_\Omega\left(\sum_{i=1}^r \hat{s}_i \hat{\lambda}_i \hat{U}_i \hat{V}_i^T - M\right)\right\|_F^2 < \left\|\mathcal{P}_\Omega\left(\sum_{i=1}^r s_{0i} \hat{\lambda}_i \hat{U}_i \hat{V}_i^T - M\right)\right\|_F^2\right) = 0,$$

where  $\hat{s}_i = \text{sign}(\langle \hat{V}_i, \mathbf{v}_i(M) \rangle) \text{sign}(\langle \hat{U}_i, \mathbf{u}_i(M) \rangle)$  and  $s_{0i} = \text{sign}(\langle \hat{V}_i, V_i \rangle) \text{sign}(\langle \hat{U}_i, U_i \rangle)$  for  $i = 1, \dots, r$ .

**Remark 4.** In Algorithm 1, when combining  $\hat{\lambda}_i$ ,  $\hat{U}_i$ , and  $\hat{V}_i^T$  to obtain  $\hat{M} = \sum_{i=1}^r \hat{s}_i \hat{\lambda}_i \hat{U}_i \hat{V}_i^T$ , there happens a sign problem. It is because  $\hat{U}_i$  and  $\hat{V}_i^T$  estimated from  $\Sigma_{t\hat{p}}$  and  $\Sigma_{\hat{p}}$  are consistent up to signs for  $U_i$  and  $V_i^T$ . So, it also estimates  $\hat{s}_i$  and Proposition 3.1 ensures that  $\hat{s}_i$  are consistent for the true sign  $s_{0i}$  for  $i = 1, \dots, r$ .



**Proposition 3.2.** (Theorem 4.4 in Cho et al. (2016)) Under Assumption 1 and the model setup in Section 2,  $\hat{M}$  is a consistent estimator of  $M_0$ . In particular,

$$\frac{1}{nd} \|\hat{M} - M_0\|_F^2 = o_p\left(\frac{h_n}{pd}\right),$$

where  $h_n$  diverges very slowly with the dimensionality, for example,  $\log(\log d)$ .

**Remark 5.** Since  $h_n$  in Proposition 3.2 can be any quantity that diverges slowly with the dimensionality, the convergence rate of  $\hat{M}$  can be thought of as  $1/pd$ . Under the setting where the rank of  $M_0$  is fixed as in this paper, it is matched to the minimax error rate,  $r/pd$ , found in Koltchinskii et al. (2011).

Using  $\hat{M}$  to initialize *Adaptive-Impute* has two major advantages. First, since  $\hat{M}$  is already a consistent estimator of  $M_0$  achieving the minimax error rate, it allows a series of the iterates of *Adaptive-Impute* coming after  $\hat{M}$  to be also consistent estimators of  $M_0$  achieving the minimax error rate (see Theorem 3.1). Second, because *Adaptive-Impute* is based on a non-convex optimization problem (see Section 4), its convergence may depend on initial values.  $\hat{M}$  provides *Adaptive-Impute* a suitable initializer.

## 3.2 Adaptive thresholds

Although the one-step consistent estimator  $\hat{M}$  achieves the minimax optimality, it does not enjoy the merit of an iterative procedure, a repeated refinement of  $\mathcal{P}_\Omega^\perp(\hat{M})$  anchored in  $\mathcal{P}_\Omega(M)$ . In this section, we propose a novel thresholding scheme with which we can refine and improve  $\mathcal{P}_\Omega^\perp(\hat{M})$  iteratively. To motivate this thresholding scheme, we first consider the case where a fully-observed noisy low-rank matrix is available. Specifically, suppose that the probability of observing each entry,  $p$ , is 1 and thus  $M_F = M_0 + \epsilon$  is observed. Under the model setup in Section 2 we can easily show that

$$\mathbb{E}(M_F^T M_F) = M_0^T M_0 + n\sigma^2 I_d \quad \text{and} \quad \mathbb{E}(M_F M_F^T) = M_0 M_0^T + d\sigma^2 I_n, \quad (2)$$

where  $I_d$  and  $I_n$  are identity matrices of size  $d$  and  $n$ , respectively. This shows that the eigenvectors of  $\mathbb{E}(M_F^T M_F)$  and  $\mathbb{E}(M_F M_F^T)$  are the same as the right and left singular vectors of  $M_0$ . Also, the top  $r$  eigenvalues of  $\mathbb{E}(M_F^T M_F)$  consist of the squared singular values of

$M_0$  and a noise,  $n\sigma^2$ , the latter of which is the same as the average of the bottom  $d - r$  eigenvalues of  $\mathbb{E}(M_F^T M_F)$ . In light of this, we want the estimator of  $M_0$  based on  $M_F$  to keep the first  $r$  singular vectors of  $M_F$  as they are, but adjust the bias occurring in the singular values of  $M_F$ . Thus, the resulting estimator is

$$\hat{M}^F = \sum_{i=1}^r \sqrt{\lambda_i^2(M_F) - \alpha} \mathbf{u}_i(M_F) \mathbf{v}_i(M_F)^T, \quad \text{where } \alpha = \frac{1}{d-r} \sum_{i=r+1}^d \lambda_i^2(M_F). \quad (3)$$

A simple extension of Proposition 3.2 shows that  $\hat{M}^F$  achieves the best possible minimax error rate of convergence,  $1/d$ , since  $p = 1$ .

Now consider the cases where a partially-observed noisy low-rank matrix  $M$  is available. For each iteration  $t \geq 1$ , we fill out the unobserved entries of  $M$  with the corresponding entries of the previous iterate  $Z_t$ , treat the completed matrix  $\widetilde{M}_t = \mathcal{P}_\Omega(M) + \mathcal{P}_\Omega^\perp(Z_t)$  as if it is a fully-observed matrix  $M_F$ , and find the next iterate  $Z_{t+1}$  in the same way that we found  $\hat{M}^F$  from  $M_F$  in (3);

$$Z_{t+1} = \sum_{i=1}^r \sqrt{\lambda_i^2(\widetilde{M}_t) - \tilde{\alpha}_t} \mathbf{u}_i(\widetilde{M}_t) \mathbf{v}_i(\widetilde{M}_t)^T, \quad \text{where } \tilde{\alpha}_t = \frac{1}{d-r} \sum_{i=r+1}^d \lambda_i^2(\widetilde{M}_t). \quad (4)$$

Note that the difference in (4) from (3) is in the usage of  $\widetilde{M}_t$  instead of  $M_F$ . Hence, the performance of *Adaptive-Impute* may depend on how close  $\mathcal{P}_\Omega(Z_t)$  is to  $\mathcal{P}_\Omega(M_0)$ .

Algorithm 2 summarizes these computing steps of *Adaptive-Impute* continued from Algorithm 1. The computational bottleneck of *Adaptive-Impute* iteration is in truncated SVD of  $\widetilde{M}_t$  and matrix multiplications for  $Z_{t+1} = \sum_{i=1}^r \lambda_i^{(t)} U_i^{(t)} V_i^{(t)T}$ . We implement this using sparse matrix representation and well developed R package, *rARPACK*, (Qiu et al. (2016)), and compute in the order  $O(dnr)$ .

The following theorem illustrates that the iterates of *Adaptive-Impute* retain the statistical performance of the initializer  $\hat{M}$ .

**Theorem 3.1.** *Under Assumption 1 and the model setup in Section 2, we have for any fixed value of  $t$ ,*

$$\frac{1}{nd} \|Z_t - M_0\|_F^2 = o_p\left(\frac{h_n}{pd}\right), \quad \text{as } n, d \rightarrow \infty \text{ with any } h_n \rightarrow \infty$$

where  $h_n$  diverges very slowly with the dimensionality, for example,  $\log(\log d)$ .

---

**Algorithm 2** *Adaptive-Impute*


---

**Require:**  $M$ ,  $y$ ,  $r$ , and  $\varepsilon > 0$

$Z_1 \leftarrow \hat{M}$  # from Algorithm 1  
**repeat** for  $t = 1, 2, \dots$   
     $\widetilde{M}_t \leftarrow \mathcal{P}_\Omega(M) + \mathcal{P}_\Omega^\perp(Z_t)$   
     $V_i^{(t)} \leftarrow \mathbf{v}_i(\widetilde{M}_t), \quad \forall i \in \{1, \dots, r\}$   
     $U_i^{(t)} \leftarrow \mathbf{u}_i(\widetilde{M}_t), \quad \forall i \in \{1, \dots, r\}$   
     $\widetilde{\alpha}_t \leftarrow \frac{1}{d-r} \sum_{i=r+1}^d \lambda_i^2(\widetilde{M}_t)$   
     $\tau_{t,i} \leftarrow \lambda_i(\widetilde{M}_t) - \sqrt{\lambda_i^2(\widetilde{M}_t) - \widetilde{\alpha}_t}, \quad \forall i \in \{1, \dots, r\}$  # Adaptive thresholds  
     $\lambda_i^{(t)} \leftarrow \lambda_i(\widetilde{M}_t) - \tau_{t,i} \left( = \sqrt{\lambda_i^2(\widetilde{M}_t) - \widetilde{\alpha}_t} \right), \quad \forall i \in \{1, \dots, r\}$   
     $Z_{t+1} \leftarrow \sum_{i=1}^r \lambda_i^{(t)} U_i^{(t)} V_i^{(t)T}$   
     $t \leftarrow t + 1$   
**until**  $\|Z_{t+1} - Z_t\|_F^2 / \|Z_t\|_F^2 \leq \varepsilon$   
**return**  $Z_{t+1}$

---

**Remark 6.** Similarly as in Remark 5, since  $h_n$  is a quantity diverging very slowly, the convergence rate of  $Z_t$  can be thought of as  $1/pd$  which is matched to the minimax error rate,  $r/pd$  (Koltchinskii et al. (2011)).

Figure 3 and Lemma 4.2 in the following sections show that the iterates of *Adaptive-Impute* get closer to the underlying low-rank matrix each iteration and the resulting estimator of *Adaptive-Impute* improves that of the one-step estimator  $\hat{M}$ , particularly for large  $n$  and  $d$ . However, since the minimax error bound that  $\hat{M}$  achieves is already a very tight bound (Proposition 3.2), Theorem 3.1 shows that the improvement that *Adaptive-Impute* made is not as much as affecting this minimax convergence rate.

### 3.3 Non-convexity of *Adaptive-Impute*

We can view *Adaptive-Impute* as an estimation method via non-convex optimization.

For  $t \geq 1$ , define

$$\tau_{t,i} = \begin{cases} \lambda_i(\widetilde{M}_t) - \sqrt{\lambda_i^2(\widetilde{M}_t) - \widetilde{\alpha}_t}, & i \leq r \\ \lambda_{r+1}(\widetilde{M}_t), & i > r \end{cases}, \quad (5)$$

where  $\tilde{\alpha}_t = \frac{1}{d-r} \sum_{i=r+1}^d \lambda_i^2(\tilde{M}_t)$  and  $\tilde{M}_t = \mathcal{P}_\Omega(M) + \mathcal{P}_\Omega^\perp(Z_t)$ . Then, in each iteration *Adaptive-Impute* provides a solution to the problem

$$\min_{Z \in \mathbb{R}^{n \times d}} \frac{1}{2nd} \|\tilde{M}_t - Z\|_F^2 + \sum_{i=1}^d \frac{\tau_{t,i}}{\sqrt{nd}} \frac{\lambda_i(Z)}{\sqrt{nd}}. \quad (6)$$

Note that the threshold parameters,  $\tau_{t,i}$ , have dependence on both the  $i$ -th singular value and the  $t$ -th iteration. The following theorem provides an explicit solution to (6).

**Theorem 3.2.** *Let  $X$  be an  $n \times d$  matrix and let  $n \geq d$ . The optimization problem*

$$\min_Z \frac{1}{2nd} \|X - Z\|_F^2 + \sum_{i=1}^d \frac{\tau_i}{\sqrt{nd}} \frac{\lambda_i(Z)}{\sqrt{nd}} \quad (7)$$

*has a solution which is given by*

$$\hat{Z} = \Phi(\Delta - \tau)_+ \Psi^T, \quad (8)$$

where  $\Phi\Delta\Psi^T$  is the SVD of  $X$ ,  $\tau = \text{diag}(\tau_1, \dots, \tau_d) \in \mathbb{R}^{d \times d}$ ,  $(\Delta - \tau)_+ = \text{diag}((\lambda_1(X) - \tau_1)_+, \dots, (\lambda_d(X) - \tau_d)_+) \in \mathbb{R}^{d \times d}$ , and  $c_+ = \max(c, 0)$  for any  $c \in \mathbb{R}$ .

**Remark 7.** *To see how Theorem 3.2 provides a solution to (6), let  $X = \tilde{M}_t$  and  $\tau_i = \tau_{t,i}$  as specified in (5). Then, (6) and (7) become the same and  $\hat{Z}$  in (8) gives the explicit form of the  $(t+1)$ -th iterate,  $Z_{t+1}$ , in Algorithm 2.*

If we set  $\tau_{t,1} = \dots = \tau_{t,r} = \tau$  and  $\tau_{t,r+1} = \dots = \tau_d = \infty$  for all  $t \geq 1$  in (6), the optimization problem (6) becomes equivalent to that of *softImpute* with a rank constraint (Mazumder et al. (2010)). While *softImpute* requires finding the right value of a thresholding parameter  $\tau$  by using a cross validation (CV) technique which is time-consuming and often does not have a straightforward validation criteria, *Adaptive-Impute* suggests specific values of the thresholding levels as in (5) which are updated according to the updated iterate in each iteration. This novel thresholding scheme of *Adaptive-Impute* together with a rank constraint results in superior empirical performances over the existing *softImpute*-type algorithms (see Section 5).

Also, connecting equation (2) to the problems (6) and (5) helps understand and improve the thresholding parameters and rank constraints that have been commonly employed in the previous matrix completion literature. As implied by (2), since the top  $r$  singular values of

$\mathcal{P}_\Omega(M) + \mathcal{P}_\Omega^\perp(Z_t)$  are mixed with signals and noise, thresholding parameters for the top  $r$  singular values reduce the singular values by the amount that we expect them to be inflated due to noise. Then, since the rest  $d - r$  singular values of  $\mathcal{P}_\Omega(M) + \mathcal{P}_\Omega^\perp(Z_t)$  are non-zero only due to the noise, we threshold them to zero with the rank constraint.

The thresholding scheme of *Adaptive-Impute* can be viewed as a solution to a non-convex optimization problem since at every iteration it differentially and adaptively thresholds the singular values. As Hastie and others alluded to a similar issue for matrix completion methods via non-convex optimization in Hastie et al. (2014), it is hard to provide a direct convergence guarantee of *Adaptive-Impute*. So, in the following section we introduce a generalized-*softImpute* algorithm, a simpler algorithm than *Adaptive-Impute* and yet still non-convex, and investigate sufficient conditions for its asymptotic convergence. It hints at the convergence behavior of *Adaptive-Impute* in the asymptotic sense by showing its similarity to generalized-*softImpute* and possibility of satisfying the conditions for convergence of generalized-*softImpute*.

## 4 Generalized *softImpute*

Generalized-*softImpute* employs differential thresholding parameters similarly to *Adaptive-Impute*, but it does not update those every iteration unlike *Adaptive-Impute*. Specifically, generalized-*softImpute* is an algorithm which iteratively solves the problem,

$$\min_{Z \in \mathbb{R}^{n \times d}} Q_\tau(Z|Z_t^g) := \frac{1}{2nd} \|\mathcal{P}_\Omega(M) + \mathcal{P}_\Omega^\perp(Z_t^g) - Z\|_F^2 + \sum_{i=1}^d \frac{\tau_i}{\sqrt{nd}} \frac{\lambda_i(Z)}{\sqrt{nd}}, \quad (9)$$

to ultimately solve the optimization problem,

$$\min_{Z \in \mathbb{R}^{n \times d}} f_\tau(Z) := \frac{1}{2nd} \|\mathcal{P}_\Omega(M) - \mathcal{P}_\Omega(Z)\|_F^2 + \sum_{i=1}^d \frac{\tau_i}{\sqrt{nd}} \frac{\lambda_i(Z)}{\sqrt{nd}}. \quad (10)$$

Note that generalized-*softImpute* differentially penalizes the singular values, but the thresholding parameters do not change over iterations. The iterative solutions of generalized-*softImpute* are denoted by  $Z_{t+1}^g := \arg \min_{Z \in \mathbb{R}^{n \times d}} Q_\tau(Z|Z_t^g)$  for  $t \geq 1$  and Theorem 3.2 provides a closed form of  $Z_{t+1}^g$ . If  $\tau_i = \tau$  for all  $1 \leq i \leq d$ , generalized-*softImpute* will be equivalent to *softImpute* and both (9) and (10) become convex problems. However, by

differentially penalizing the singular values, generalized-*softImpute* ends up solving a non-convex optimization problem. Theorem 4.1 below shows that despite the non-convexity of generalized-*softImpute*, the iterates of generalized-*softImpute*,  $\{Z_t^g\}_{t \geq 1}$ , converge to a solution of problem (10) under certain conditions.

**Assumption 2.** Let  $\widetilde{M}_t^g = \mathcal{P}_\Omega(M) + \mathcal{P}_\Omega^\perp(Z_t^g)$  and  $D_t^g := \widetilde{M}_t^g - Z_{t+1}^g$ . Then,

$$\frac{1}{nd} \|D_t^g - D_{t+1}^g\|_F^2 + \frac{2}{nd} \langle D_t^g - D_{t+1}^g, Z_{t+1}^g - Z_{t+2}^g \rangle \geq 0 \quad \text{for all } t \geq 1.$$

**Theorem 4.1.** Let  $Z_\infty$  be a limit point of the sequence  $Z_t^g$ . Under Assumption 2, if the minimizer  $Z^s$  of (10) satisfies

$$Z^s \in \left\{ Z \in \mathbb{R}^{n \times d} : \sum_{i=1}^d \tau_i \lambda_i(Z) \geq \sum_{i=1}^d \tau_i \lambda_i(Z_\infty) + \langle (Z - Z_\infty), D_\infty \rangle \right\}, \quad (11)$$

we have  $f_\tau(Z_\infty) = f_\tau(Z^s)$  and  $\lim_{t \rightarrow \infty} f_\tau(Z_t^g) = f_\tau(Z^s)$ .

**Remark 8.** Assumption 2 and (11) are related to the sub-gradient of  $\|Z_{t+1}^g\|_*$ . Specifically, if  $\tau_i = \tau$  for all  $i$  as in *softImpute* so that the penalty  $\|Z_{t+1}^g\|_*$  is convex, those conditions are not needed since  $\frac{1}{\tau} D_t^g$  belongs to the sub-gradient of  $\|Z_{t+1}^g\|_*$  and so the conditions are always satisfied. However, in case of generalized-*softImpute* which uses a non-convex penalty, we require the two conditions to be satisfied.

**Remark 9.** If  $Z^s$  is unique, then generalized-*softImpute* finds the global minimum point of (10) by Theorem 4.1.

Generalized-*softImpute* resembles *Adaptive-Impute* in a sense that both of them employ different thresholding parameters on  $\lambda_i(Z)$ 's. However, *Adaptive-Impute* updates these tuning parameters every iteration while generalized-*softImpute* does not. The following lemmas show that despite this difference, the convergence behavior of *Adaptive-Impute* is asymptotically close to that of generalized-*softImpute*.

**Lemma 4.1.** Under Assumption 1 and the model setup in Section 2, we have

$$\left| \frac{\tau_{t,i}}{\sqrt{nd}} - \frac{\tau_{t+1,i}}{\sqrt{nd}} \right| = o_p \left( \sqrt{\frac{h_n}{pd}} \right) \quad \text{for } i = 1, \dots, d,$$

where  $\tau_{t,i}$  is defined in (5).

**Lemma 4.2.** Let  $D_t := \widetilde{M}_t - Z_{t+1}$ , where  $\widetilde{M}_t$  and  $Z_t$  are as defined in Algorithm 2. Then, under Assumption 1 and the model setup in Section 2, we have

$$\frac{1}{nd} \|D_t - D_{t+1}\|_F^2 + \frac{2}{nd} \langle D_t - D_{t+1}, Z_{t+1} - Z_{t+2} \rangle + o_p\left(\frac{h_n}{pd}\right) \geq 0.$$

Lemma 4.1 shows that for large  $n$  and  $d$ , thresholding parameters of *Adaptive-Impute* are stable between iterations so that *Adaptive-Impute* behaves similarly to generalized-*softImpute*. Lemma 4.2 shows how Assumption 2 is adapted in *Adaptive-Impute*. It implies a possibility of *Adaptive-Impute* satisfying Assumption 2 asymptotically. Although this still does not provide a direct guarantee of convergence of *Adaptive-Impute*, numerical results in Section 5 below strongly support this possibility.

We introduced generalized-*softImpute* in this section to help understand the asymptotic convergence behavior of *Adaptive-Impute*. Implementing generalized-*softImpute* has a serious problem in practice since it requires  $d$  number of thresholding parameters to be tuned. One may implement generalized-*softImpute* by using the thresholding parameters that *Adaptive-Impute* would use in the first iteration throughout the iterations. However, in this case, the performance of generalized-*softImpute* may be worse than that of *Adaptive-Impute* which updates the thresholding parameters every iteration.

## 5 Numerical results

In this section, we conducted simulations and a data analysis to compare *Adaptive-Impute* for estimating  $M_0$  with the four different versions of *softImpute*:

1. *Adaptive-Impute*: the proposed algorithm, as summarized in Algorithm 2;
2. *softImpute*: the original *softImpute* algorithm (Mazumder et al. (2010));
3. *softImpute-Rank*: *softImpute* with rank restriction (Hastie et al. (2014));
4. *softImpute-ALS*: *Maximum-Margin Matrix Factorization* (Hastie et al. (2014));
5. *softImpute-ALS-Rank*: *rank-restricted Maximum-Margin Matrix Factorization* in Algorithm 3.1 (Hastie et al. (2014)).

All experiments and analyses were done in R R Core Team (2017). *SoftImpute* algorithms were implemented with the R package, `softImpute` (Hastie and Mazumder (2015)). The R code for *Adaptive-Impute* is available at <https://github.com/chojuhee/adaptiveImpute>. In this R code, we made two adjustments from Algorithms 1 and 2 for technical reasons. First, in many of real world applications that need matrix completion, the entries of  $M_0$  are bounded below and above by constants  $L_1$  and  $L_2$  such that

$$L_1 \leq M_{0ij} \leq L_2$$

and smaller or larger values than the constants do not make sense. So, after each iteration of *Adaptive-Impute*,  $t \geq 1$ , we replace the values of  $Z_t$  that are smaller than  $L_1$  with  $L_1$  and the values of  $Z_t$  that are greater than  $L_2$  with  $L_2$ . When the exact values of  $L_1$  and  $L_2$  are not clear, one can set them to be small and large constants, respectively. When their values are clearly known, providing the exact values of them will result in a better prediction. We investigate this further in Section 5.2 with a data example. Second, the cardinality of the set,  $\{-1, 1\}^r$ , that we search over to find  $\hat{s}$  in Algorithm 1 increases exponentially. Hence, finding  $\hat{s}$  easily becomes a computational bottleneck of *Adaptive-Impute* or is even impossible for large  $r$ . We suggest two possible solutions to this problem. One solution is to find  $\hat{s}$  by computing  $\hat{s}_i = \text{sign}(\langle \hat{V}_i, \mathbf{v}_i(M) \rangle) \text{sign}(\langle \hat{U}_i, \mathbf{u}_i(M) \rangle)$  for  $i = 1, \dots, r$ . Note that if we use  $V_i$  and  $U_i$  instead of  $\mathbf{v}_i(M)$  and  $\mathbf{u}_i(M)$ , this gives us the true sign  $s_0$  under Assumption 1. The other solution is to use a linear regression. Let a vector of the observed entries of  $M$  be the dependent variable and let a vector of the corresponding entries of  $\hat{\lambda}_i \hat{U}_i \hat{V}_i^T$  be the  $i$ -th column of the design matrix for  $i = 1, \dots, r$ . Then, we set  $\hat{s}$  to be the coefficients of the regression line whose intercept is forced to be 0. The difference in the results of these two methods are negligible. In the following experiment, we only reported the results of the former solution for simplicity, while the R code provided in <https://github.com/chojuhee/adaptiveImpute> are written for both solutions. *Adaptive-Impute* is implemented with `svds()` and `Matrix()` functions from R packages “rARPACK (Qiu et al. (2016))” and “Matrix (Bates and Maechler (2017))”, respectively.



## 5.1 Simulation study

To create  $M_0 = AB^T \in \mathbb{R}^{n \times d}$ , we sampled  $A \in \mathbb{R}^{n \times r}$  and  $B \in \mathbb{R}^{d \times r}$  to contain i.i.d. uniform $[-5, 5]$  random variables and a noise matrix  $\epsilon \in \mathbb{R}^{n \times d}$  to contain i.i.d.  $\mathcal{N}(0, \sigma^2)$ . Then, each entry of  $M_0 + \epsilon$  was observed independently with probability  $p$ . Across simulations,  $n = 1700$ ,  $d = 1000$ ,  $r \in \{5, 10, 20, 50\}$ ,  $\sigma$  varies from 0.1 to 50, and  $p$  varies from 0.1 to 0.9. For each simulation setting, the data was sampled 100 times and the errors were averaged.

To evaluate performance of the algorithms, we measured three different types of errors; test, training, and total errors; the test error,  $\text{Test}(\hat{M}) = \|\mathcal{P}_\Omega^\perp(\hat{M} - M_0)\|_F^2 / \|\mathcal{P}_\Omega^\perp(M_0)\|_F^2$ , represents the distance between the estimate  $\hat{M}$  and the parameter  $M_0$  measured on the unobserved entries, the training error,  $\text{Training}(\hat{M}) = \|\mathcal{P}_\Omega(\hat{M} - M_0)\|_F^2 / \|\mathcal{P}_\Omega(M_0)\|_F^2$ , the distance measured on the observed entries, and the total error,  $\text{Total}(\hat{M}) = \|\hat{M} - M_0\|_F^2 / \|M_0\|_F^2$ , the distance measured on all entries. For ease of comparison, Figure 1 and 2 plot the relative efficiencies with respect to *softImpute*-Rank. For example, the relative test efficiency of *Adaptive-Impute* with respect to *softImpute*-Rank is defined as  $\text{Test}(\hat{M}_{rank}) / \text{Test}(\hat{M}_{adapt})$ , where  $\hat{M}_{adapt}$  is an estimate of *Adaptive-Impute* and  $\hat{M}_{rank}$  is an estimate of *softImpute-Rank*. The relative total and training efficiencies with respect to *softImpute*-Rank are defined similarly. Functions from R packages “ggplot2 (Wickham et al. (2018))” and “dplyr (Wickham et al. (2018))” were used to for all figures in this section.

We used the optimal tuning parameter for all algorithms in comparison. Specifically, for algorithms with rank restriction (including *Adaptive-Impute*), we provided the true rank values (i.e. 5, 10, 20, or 50). Although *Adaptive-Impute* does not require any other tuning parameter, for *softImpute*-type algorithms which requires to tune thresholding parameters, we provided an oracle thresholding parameter minimizing the total errors.

Figure 1 shows the change of the relative efficiencies as the probability of observing each entry,  $p$ , increases with  $\sigma = 1$ . Three columns of plots in Figure 1 correspond to three different types of errors and four rows of plots to four different values of the rank. In all cases, *Adaptive-Impute* outperforms the competitors and works especially better when  $p$  is small. Among *softImpute*-type algorithms, the algorithms with rank constraint (i.e.

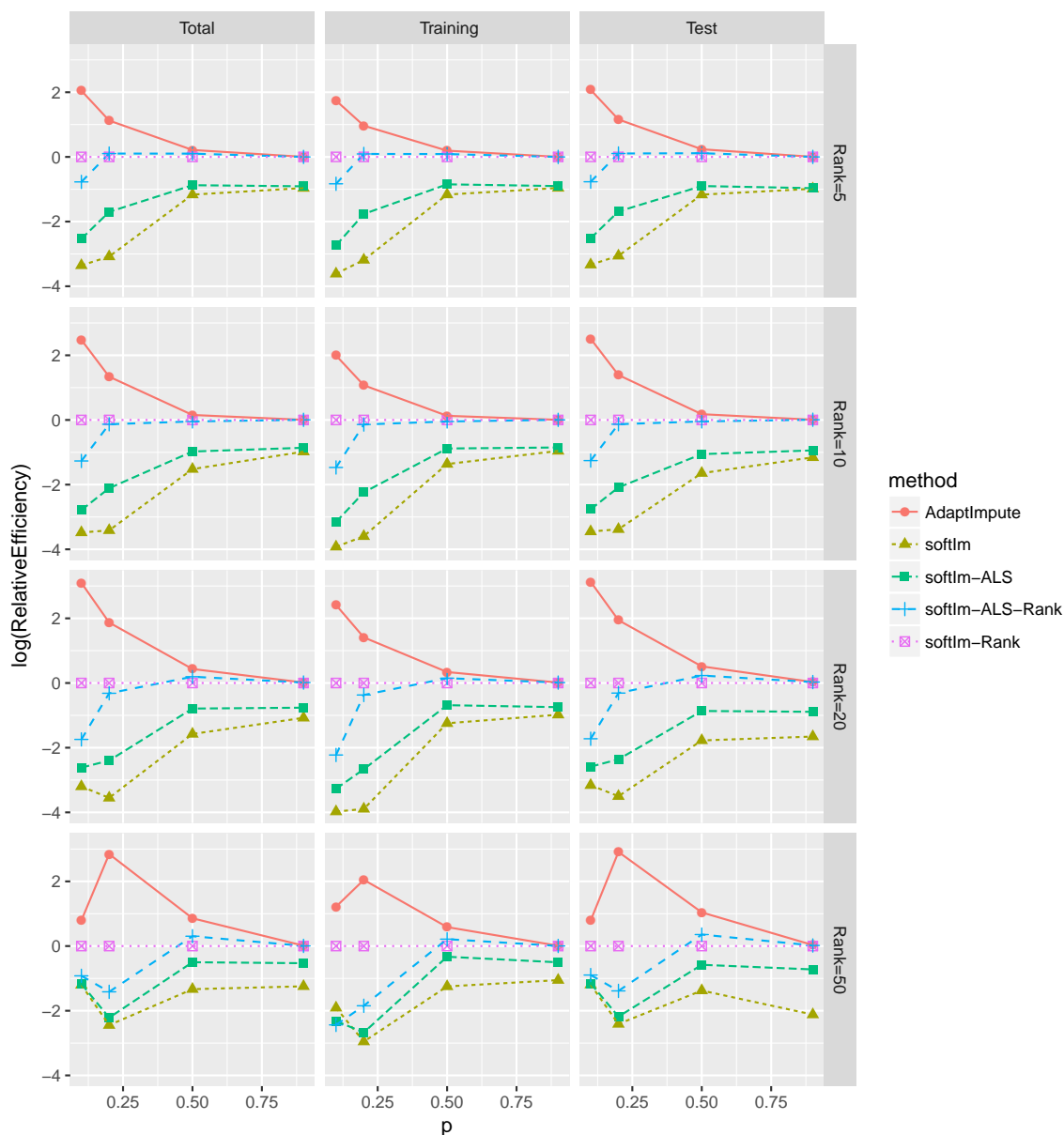


Figure 1: It shows the log relative efficiency with respect to *softImpute*-Rank plotted against the probability of observing each entry,  $p$ , when  $\sigma = 1$ . Training errors are measured over the observed entries, test errors over the unobserved entries, and total errors over all entries. *Adaptive-Impute* outperforms all other methods especially when  $p$  is small. Among *softImpute*-type algorithms, the algorithms with rank constraint perform better than the ones without.

*softImpute*-Rank and *softImpute*-ALS-Rank) perform better than the ones without (i.e. *softImpute* and *softImpute*-ALS). Figure 2 shows the change of the log relative efficiencies as the standard deviation (SD) of each entry of  $\epsilon$ ,  $\sigma$ , increases with  $p = 0.1$ . When the noise level is under 15, *Adaptive-Impute* outperforms the competitors, but when the noise level is over 15, *softImpute*-type algorithms start to outperform *Adaptive-Impute*. Hence, *softImpute*-type algorithms are more robust to large noises than *Adaptive-Impute*. It may be because when there exist large noises dominating the signals, the conditions for convergence presented in Section 4 are not satisfied. In real life applications, however, it is not common to observe such large noises that dominate the signals. The change of the absolute errors that correspond to Figure 1 and 2 are presented in Section S2 in the Supplement.

Figure 3 shows convergence of the iterates of *Adaptive-Impute* to the underlying low-rank matrix over iterations; that is, the change of  $\log \text{Total}(Z_t)$ ,  $\text{Training}(Z_t)$ , and  $\text{Test}(Z_t)$  errors as  $t$  increases. Across all plots,  $n = 1700$ ,  $d = 1000$ ,  $p = 0.1$ , and the errors were averaged over 100 replicates. In all cases, we observe that *Adaptive-Impute* converges well. Particularly, the smaller value of noise and/or rank is, the faster *Adaptive-Impute* converges.

To check the effect of the correlation structure of  $M_0$ , we generated  $M_0 = AB^T \in \mathbb{R}^{n \times d}$  as follows. We sampled  $A = (A_1^T, \dots, A_n^T)$  and  $B = (B_1^T, \dots, B_d^T)$ , where  $A_i, B_j \in \mathbb{R}^r, i = 1, \dots, n, j = 1, \dots, d$ , are i.i.d.  $\mathcal{N}(0, \rho J_r + (1 - \rho)I_r)$ , and  $I_r$  and  $J_r$  are  $r$ -dimensional identity and all-ones matrices, respectively. Then using the same scheme in the previous simulation, we generated partially observed noisy matrices. In this simulation, we fixed  $n = 1700$ ,  $d = 1000$ ,  $r = 5$ ,  $\sigma^2 = 1$  and  $p = 0.1$ , and  $\rho$  varied from 0 to 0.7. The tuning parameters were selected in the same way as the previous simulation. For each simulation setting, the data were sampled 100 times and the errors were averaged.

Figure 4 plots the total, training, and test log errors against the correlation  $\rho$ . The result is similar to the previous simulation study. That is, we find that *Adaptive-Impute* shows the best performance. One of the interesting findings is that except the *softImpute*-rank, the errors tend to decrease as the correlation increases. This may be because as the correlation increases, the observed entries contain more information about the missing entries.

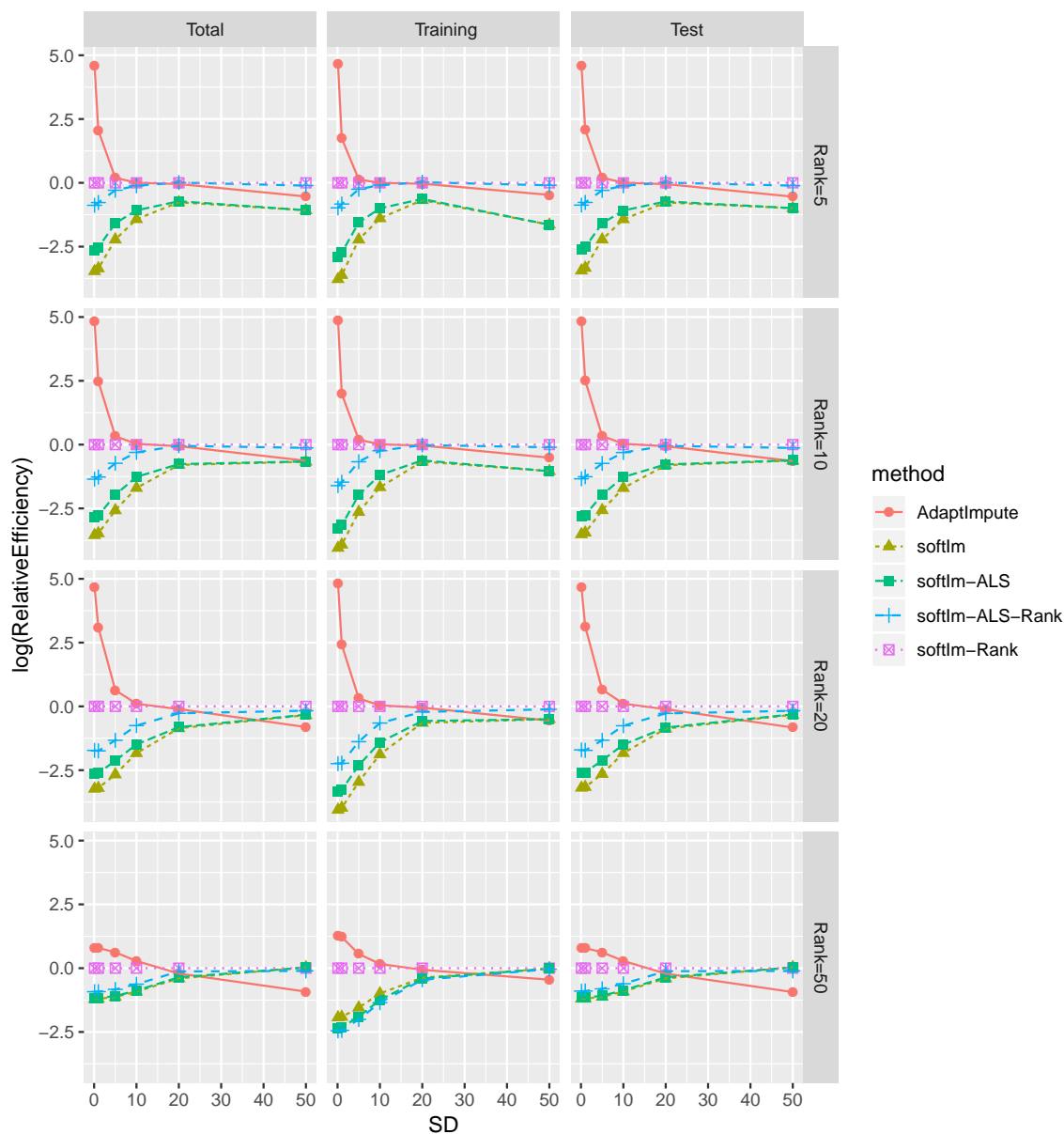


Figure 2: It shows the log relative efficiency with respect to *softImpute*-Rank plotted against the SD of each entry of  $\epsilon$  when  $p = 0.1$ . Training errors are measured over the observed entries, test errors over the unobserved entries, and total errors over all entries. When the noise level is under 15, *Adaptive-Impute* outperforms the competitors, but when the noise level is over 15, *softImpute*-type algorithms start to outperform *Adaptive-Impute*. Overall, *softImpute*-type algorithms are more robust to large noises than *Adaptive-Impute*.

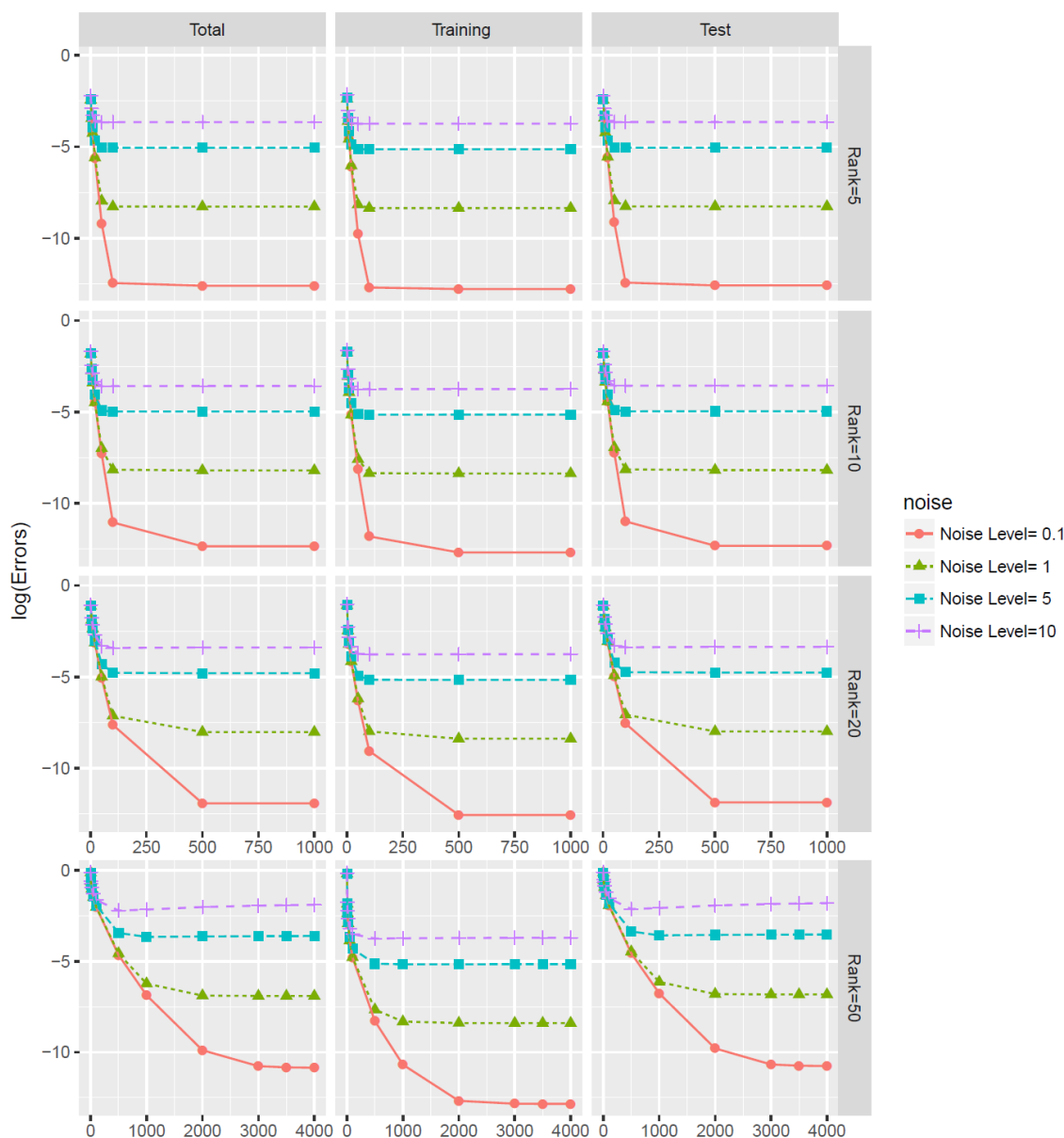


Figure 3: It shows convergence of the iterates of *Adaptive-Impute* to the underlying low-rank matrix. In all plots,  $n = 1700$ ,  $d = 1000$ ,  $p = 0.1$ , and all points were averaged over 100 replicates. We observe that *Adaptive-Impute* converges well in all cases. Particularly, the smaller value of noise and/or rank is, the faster *Adaptive-Impute* converges.

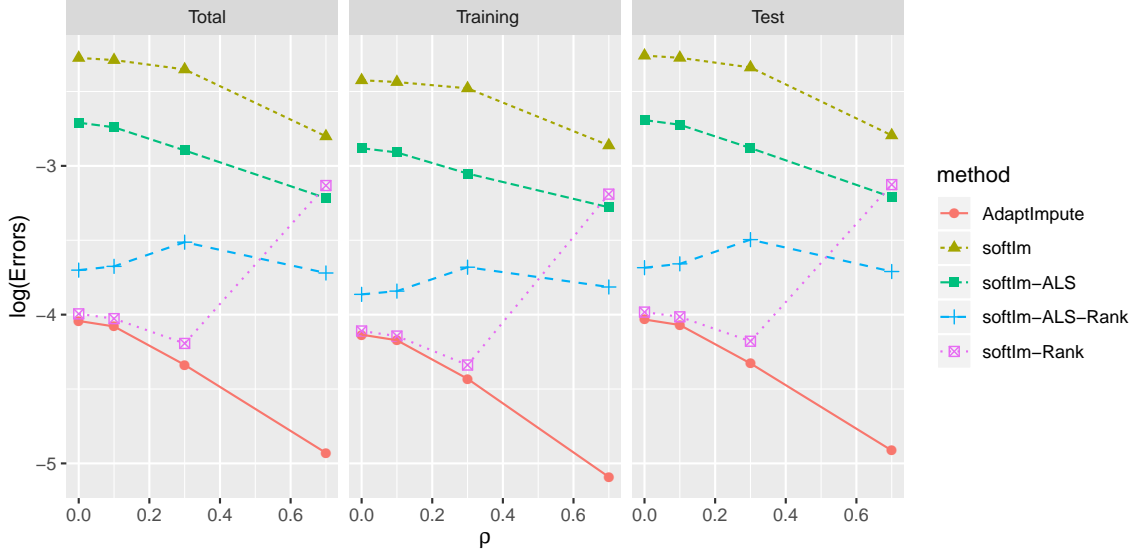


Figure 4: It shows the change of the log errors when the correlation  $\rho$  increases and  $\sigma = 1, p = 0.1, n = 1700, d = 1000$ . *Adaptive-Impute* outperforms the competitors. As  $\rho$  increases, the observed entries contain more information about the missing entries and so the errors tend to decrease as the correlation increases except for the errors of *softImpute-rank*.

## 5.2 A data example

We applied *Adaptive-Impute* and the competing methods to the MovieLens 100k (GroupLens (2015)) data. We used 5 training and 5 test data sets from 5-fold CV which are publicly available in GroupLens (2015). To tune the rank and/or the thresholding parameters, we first chose  $r = 3$  for *Adaptive-Impute* based on a scree plot (Figure 5). Lemma 2 in Cho et al. (2016) provides justification of using the scree plot and the singular value gap to choose the rank. For *SoftImpute*-type algorithms, we chose the optimal value of  $\tau$  minimizing test errors in cases where there is no rank constraint and chose the optimal set of values of  $(\tau, r)$  minimizing test errors in cases where there is a rank constraint. The test errors were measured by normalized mean absolute error (NMAE) (Herlocker et al. (2004)),

$$\frac{1}{(M_{max} - M_{min})|\Omega_{test}|} \sum_{(i,j) \in \Omega_{test}} |\hat{M}_{ij} - M_{ij}|,$$

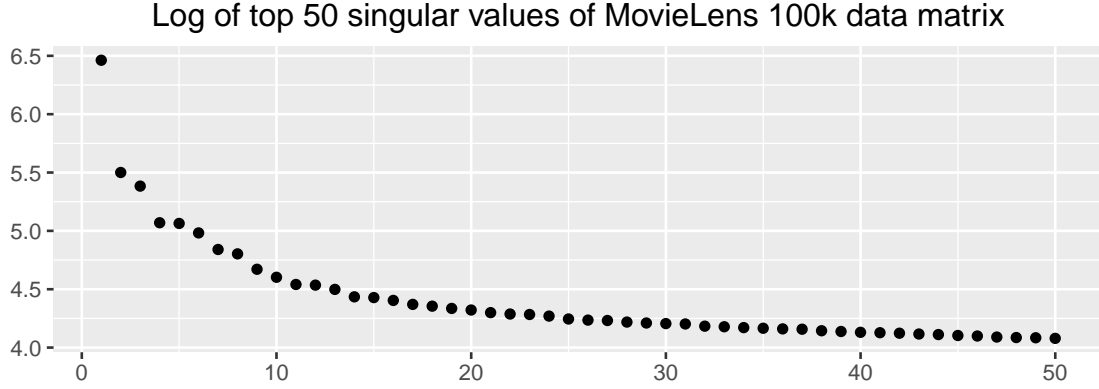


Figure 5: Log of the top 50 singular values of the MovieLens 100k data matrix (GroupLens (2015)).

and by mean squared error (MSE),

$$\frac{1}{|\Omega_{test}|} \sum_{(i,j) \in \Omega_{test}} (\hat{M}_{ij} - M_{ij})^2,$$

where the set  $\Omega_{test}$  contains indices of the entries in test data,  $|\Omega_{test}|$  is the cardinality of  $\Omega_{test}$ ,  $M_{max} = \max\{\{M_{i,j}\} \setminus 0\}$  is the largest entry of  $M$ , and  $M_{min} = \min\{\{M_{i,j}\} \setminus 0\}$  is the smallest entry of  $M$ .

Table 1 summarizes the resulting NMAEs. Columns correspond to the 5-fold CV test data and rows represent the 5 different algorithms in comparison. We observe that *Adaptive-Impute* outperforms all of the other algorithms. Specifically, the test errors of *Adaptive-Impute* measured in NMAE reduce those of *softImpute*-type algorithms by 30%-34%. Among *softImpute*-type algorithms, the ones with rank constraint (i.e. *softImpute-Rank* and *softImpute-ALS-Rank*) performs better than the ones without (i.e. *softImpute* and *softImpute-ALS*). This is the same result to the simulation results. We can observe similar results as above with MSEs. The summary of the resulting MSEs is in Table 1 in the Supplement.

We tested *Adaptive-Impute* to the MovieLens 100k (GroupLens (2015)) data also to see how robust it is to the accuracy of  $L_1$  and  $L_2$ . The dataset describes 5-star ratings and in the analyses above  $L_1$  and  $L_2$  were set to be 1 and 5, respectively. In Table 2 we changed  $L_1$  and  $L_2$  to be  $1 - \alpha$  and  $5 + \alpha$  for  $\alpha = 0, 1, 5, 10, 50, 100$ , and  $\infty$ , and summarized the

Table 1: The NMAEs of *Adaptive-Impute* and its competitors measured in 5-fold CV test data from MovieLens 100k (GroupLens (2015)). *Adaptive-Impute* outperforms the competitors in all cases.

	test data 1	test data 2	test data 3	test data 4	test data 5
<i>Adaptive-Impute</i>	0.1837	0.1826	0.1820	0.1822	0.1845
<i>softImpute</i>	0.5909	0.5910	0.5840	0.5811	0.5874
<i>softImpute</i> -Rank	0.5722	0.5454	0.5361	0.5392	0.5504
<i>softImpute</i> -ALS	0.5910	0.5910	0.5840	0.5811	0.5873
<i>softImpute</i> -ALS-Rank	0.5722	0.5454	0.5361	0.5392	0.5504

the resulting NMAEs. The test errors of *Adaptive-Impute* measured in NMAE tended to increase as the bound by  $L_1$  and  $L_2$  gets loose. However, the changes were not dramatic and the NMAEs increased only by 1%-5%. We obtained similar results with MSEs. The summary of the resulting MSEs is in Table 2 in the Supplement.

Table 2: The NMAEs of *Adaptive-Impute* change as  $L_1 = 1 - \alpha$  and  $L_2 = 5 + \alpha$  change for  $\alpha = 0, 1, 5, 10, 50, 100$ , and  $\infty$ . The NMAEs were measured in 5-fold CV test data from MovieLens 100k (GroupLens (2015)). The test errors tends to increase as the bound by  $L_1$  and  $L_2$  gets loose. However, the changes are not dramatic

	test data 1	test data 2	test data 3	test data 4	test data 5
0	0.1837	0.1826	0.1820	0.1822	0.1845
1	0.1872	0.1847	0.1844	0.1846	0.1864
5	0.1892	0.1868	0.1861	0.1870	0.1892
10	0.1882	0.1855	0.1860	0.1875	0.1891
50	0.1924	0.1896	0.1903	0.1861	0.1859
100	0.1923	0.1900	0.1908	0.1904	0.1913
$\infty$	0.1923	0.1901	0.1908	0.1904	0.1913



## 6 Discussion

Choosing the right thresholding parameter for matrix completion algorithms using thresholded SVD often poses empirical challenges. This paper proposed a novel thresholded SVD algorithm for matrix completion, *Adaptive-Impute*, which employs a theoretically-justified and data-dependent set of thresholding parameters. We established its theoretical guarantees on statistical performance and showed its strong performances in both simulated and real-world data. It provides understanding on the effects of thresholding and the right threshold level. Yet, there is a newly open problem. Although we proposed a reasonable remedy in the paper, the choice of the rank of the underlying low-rank matrix is of another great practical interest. To estimate the rank and completely automate the entire procedure of *Adaptive-Impute* would be a potential direction for future research.

## References

- Bates, D. and M. Maechler (2017). Matrix: Sparse and dense matrix classes and methods @MISC. <https://CRAN.R-project.org/package=Matrix>.
- Cai, J.-F., E. J. Candès, and Z. Shen (2010). A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization* 20(4), 1956–1982.
- Cai, T. and W. Liu (2011). Adaptive thresholding for sparse covariance matrix estimation. *Journal of the American Statistical Association* 106(494), 672–684.
- Candès, E. J. and Y. Plan (2010). Matrix completion with noise. *Proceedings of the IEEE* 98(6), 925–936.
- Chatterjee, S. (2014). Matrix estimation by universal singular value thresholding. *The Annals of Statistics* 43(1), 177–214.
- Cho, J., D. Kim, and K. Rohe (2016). Asymptotic theory for estimating the singular vectors and values of a partially-observed low rank matrix with noise. *Statistica Sinica*, To be appeared.

- Donoho, D. and M. Gavish (2014). Minimax risk of matrix denoising by singular value thresholding. *The Annals of Statistics* 42(6), 2413–2440.
- Fan, J., Y. Liao, and M. Mincheva (2013). Large covariance estimation by thresholding principal orthogonal complements. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 75(4), 603–680.
- Gavish, M. and D. L. Donoho (2014). The optimal hard threshold for singular values is  $4/\sqrt{3}$ . *IEEE Transactions on Information Theory* 60(8), 5040–5053.
- GroupLens (2015). Movielens100k @MISC. <http://grouplens.org/datasets/movielens/>.
- Hastie, T. and R. Mazumder (2015). softimpute @MISC. <https://cran.r-project.org/web/packages/softImpute/index.html>.
- Hastie, T., R. Mazumder, J. Lee, and R. Zadeh (2014). Matrix completion and low-rank svd via fast alternating least squares. *arXiv preprint arXiv:1410.2596*.
- Herlocker, J. L., J. A. Konstan, L. G. Terveen, and J. T. Riedl (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)* 22(1), 5–53.
- Horn, R. A. and C. R. Johnson (1990). *Matrix analysis*. Cambridge university press.
- Keshavan, R., A. Montanari, and S. Oh (2009). Matrix completion from noisy entries. In *Advances in Neural Information Processing Systems*, pp. 952–960.
- Koltchinskii, V., K. Lounici, and A. B. Tsybakov (2011). Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics* 39(5), 2302–2329.
- Mazumder, R., T. Hastie, and R. Tibshirani (2010). Spectral regularization algorithms for learning large incomplete matrices. *The Journal of Machine Learning Research* 11, 2287–2322.

- Negahban, S. and M. J. Wainwright (2011). Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics* 39(2), 1069–1097.
- Qiu, Y., J. Mei, and authors of the ARPACK library. See file AUTHORS for details. (2016). rARPACK: Solvers for large scale eigenvalue and SVD problems @MISC. <https://cran.r-project.org/web/packages/rARPACK/index.html>.
- R Core Team (2017). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing.
- Rennie, J. D. and N. Srebro (2005). Fast maximum margin matrix factorization for collaborative prediction. In *Proceedings of the 22nd international conference on Machine learning*, pp. 713–719. ACM.
- Wickham, H., W. Chang, L. Henry, T. L. Pedersen, K. Takahashi, C. Wilke, and K. Woo (2018). ggplot2: Create elegant data visualisations using the grammar of graphics @MISC. <https://cran.r-project.org/package=ggplot2>.
- Wickham, H., R. François, L. Henry, and K. Müller (2018). dplyr: A grammar of data manipulation @MISC. <https://cran.r-project.org/package=dplyr>.
- Zou, H. (2006). The adaptive lasso and its oracle properties. *Journal of the American statistical association* 101(476), 1418–1429.