Fixed Point Iteration Algorithms for Low-rank Matrix Completion

by

Xingliang Huang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Statistics

Waterloo, Ontario, Canada, 2015

Abstract

A lot of applications can be formulated as matrix completion problems. In order to address such problems, a common assumption is that the underlying matrix is (approximately) low-rank. Under certain conditions, the recovery of low-rank matrix can be done via nuclear norm minimization, a convex program.

Scalable and fast algorithms are essential as the practical matrix completion tasks always occur on a large scale. Here we study two algorithms and generalize the unified framework of fixed point iteration algorithm. We derive the convergence results and propose a new algorithm based on the insights. Compared with the baseline algorithms, our proposed method is significantly more efficient without loss of precision and acceleration potentiality.

Acknowledgements

I cherish the memory of writing the thesis. I would like to thank all the people who made this possible.

I thank my supervisor, Professor Mu Zhu, for his kindly support and guidance. When I struggled to change the topic, it is Mu's encouragement that opened a new door for me. His brilliant idea and critical thinking inspired me a lot.

I thank Ms. Mary Lou Dufton for her perfect coordination job, and all the faculty members who have ever taught or worked with me.

I thank my family and friends, I could not imagine my life without your company.

Dedication

This is dedicated to the one I love.

Table of Contents

List of Tables **viii**

List of Figures **ix**

1 Introduction **1**

1.1 Nuclear Norm Heuristic 1

1.2 Review of Algorithms 2

1.3 Organization 3

2 Low-rank Matrix Completion via Nuclear Norm Minimization **5**

2.1 Motivation 5

2.2 Problem Formulation 7

2.3 SVT as Minimizer 8

2.4 Non-expansiveness of SVT 9

2.5 Relationship with the Lasso 11

2.6 Relationship with Matrix Factorization 12

3 Fixed Point Iteration Algorithm **15**

3.1 Soft-Impute . . . 15

3.2 Fixed Point Iteration . . . 16

3.3 Convergence Property . . . 18

3.4 Convergence Rate . . . 22

3.5 Selection of τ . . . 27

3.6 New Algorithms . . . 29

3.7 Other Issues . . . 31

4 Numerical Experiments **34**

4.1 Settings and Main Results . . . 34

4.2 Simulation Study . . . 35

4.3 Acceleration Result . . . 45

4.4 An Image Example . . . 48

5 Conclusion and Future Work **56**

Appendix A Convex Analysis **58**

Appendix B SDP Form of Nuclear Norm Minimization **60**

References **62**

List of Tables

4.1 Parameters of randomly generated matrices 37

4.2 Numerical results for randomly generated matrix completion with noise free observations, where $\gamma = \frac{2}{3}$, $\mu = \sqrt{m}$ 38

4.3 Numerical results for randomly generated matrix completion with noisy observations, where SNR $= 9$, $\gamma = \frac{2}{3}$; $\mu = \sqrt{m}$ for $m = 100, 200, 500$; $\mu = 1.5\sqrt{m}$ for $m = 1000$ 39

4.4 Numerical results for randomly generated matrix completion with noisy observations, where SNR $= 6$, $\gamma = \frac{2}{3}$; $\mu = \sqrt{m}$ for $m = 100, 200, 500$; $\mu = 1.5\sqrt{m}$ for $m = 1000$ 40

4.5 Number of formal iterations and equivalent iterations, where $m = 1000$, $r = 50$, SR $= 25\%$, SNR $= 9$, $\mu = 1.5\sqrt{m}$ 44

4.6 Running time of FPI1 and AFPI embedded with `PROPACK`. The examples are from the last rows of Table 4.2 – 4.4, where $m = 1000$, rank $= 50$, SR $= 25\%$, $\mu = \sqrt{m}$ for noiseless case; $\mu = 1.5\sqrt{m}$ for SNR $= 6$ and SNR $= 9$. 46

4.7 Records of image recovery, where 50% pixels masked uniformly at random and the true rank is 768. Set $\mu = 1.25 \times \sqrt{1024}$, tol $= 1 \times 10^{-3}$ 48

List of Figures

4.1 Change of τ over formal iterations, where $m = 1000$, $r = 50$, SR $= 25\%$, SNR $= 9$, $\mu = 1.5\sqrt{m}$ 41

4.2 Change of $f_\mu(X)$ over formal iterations, where $m = 1000$, $r = 50$, SR $=$ 25%, SNR $= 9$, $\mu = 1.5\sqrt{m}$ 42

4.3 Part of Figure 4.2, highlight the difference between FPI2, FPILS and AFPI from iteration 10 to 20 43

4.4 Change of rank over iterations, where $m = 1000$, $r = 50$, SR $= 25\%$, $\mu = \sqrt{m}$ and noiseless 47

4.5 Original Figure (768×1024), rank $= 768$ 49

4.6 Remove 50% pixels of the original figure uniformly at random 50

4.7 Figure recovered by FPI1, rank $= 335$, error $= 4.46 \times 10^3$ 51

4.8 Figure recovered by FPI2, rank $= 317$, error $= 4.29 \times 10^3$ 52

4.9 Figure recovered by FPILS, rank $= 315$, error $= 4.25 \times 10^3$ 53

4.10 Figure recovered by AFPI, rank $= 312$, error $= 4.26 \times 10^3$ 54

Chapter 1

Introduction

1.1 Nuclear Norm Heuristic

Seeking sparse representation is helpful when dealing with high dimensional data. For data in matrix form, the concept of sparsity is equivalent to low rank. With the low rank setting, we are able to solve the matrix completion problem, i.e., to recover a low-rank matrix with partial observations. The problem arises in plenty of applications, including machine learning, bioinformatics, computer vision etc.

Low-rank matrix completion is a special case of affine rank minimization problem, which is of the following form,

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \operatorname{rank}(X) \\ \text{subject to} \quad & \mathcal{A}(X) = b, \end{aligned} \tag{1.1}$$

where $X \in \mathbb{R}^{m \times n}$ and $\mathcal{A}(X) = b$ denotes the affine restrictions on X. For matrix completion problem, the observation can be written in this affine formulation.

Nevertheless, solving (1.1) directly is prohibitively challenging, since it is non-convex and NP-hard in general. Fazel et al. (2001) suggest that the nuclear norm , i.e., the sum of all the singular values, is the tightest convex surrogate of rank. Meanwhile, Fazel et al. figure out that the nuclear norm minimization problem (1.2) could be formulated as a *semidefinite programming* (SDP).

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \|X\|_* \\ \text{subject to} \quad & \mathcal{A}(X) = b. \end{aligned} \tag{1.2}$$

As a convex optimization problem, (1.2) has a global minimum and is much easier to handle compared with (1.1). It is a natural thinking to approximate (1.1) with (1.2), which is called the nuclear norm heuristic.

A number of studies have shown that (1.1) and (1.2) can be ”formally equivalent” given certain conditions, i.e., both problems have a unique, identical solution. The first substantial work in this direction was done by Recht et al. (2010). They propose several restricted isometry conditions to establish the equivalence. Under a different framework, Candès and Recht (2009) construct the equivalence based on the incoherence structure of the matrices. Their work is further improved by Candès and Tao (2010); Gross (2011); Recht (2011) in various aspects, respectively.

The above work consider only the noiseless case, i.e., the observed entries are exactly from a low-rank matrix without noise. Candès and Plan (2010) claim that nuclear norm minimization performs stably to small, elementwise noise in observations, and the recovery error is proportional to the noise level.

1.2 Review of Algorithms

As we mentioned, (1.2) can be formulated as an SDP (see Lemma B.2). SDP is a significant class of convex optimization problems, and there are many off-the-shelf solvers, e.g., `cvx` (Grant and Boyd, 2014). These implementations are mainly based on the famous interior-point paradigm (e.g., Vandenberghe and Boyd, 1996). On moderate size examples, they perform well with high accuracy and fast convergence rate. However, the interior-point method relies on second-order information, thus it is not scalable due to the storage and computation complexity. Empirical result presents that on a common personal computer, these SDP solvers can handle a 100×100 matrix at most (Lin et al., 2010). In practice, low-rank matrix completion tasks always appear with extremely large matrices, for instance, the famous *Netflix Prize* has a user-item rating matrix of size $10^8 \times 10^5$.

Aware of the scalability weakness of the interior-point method, the studies on the first-order algorithms grows rapidly. The *singular value thresholding* (SVT) operator, which is introduced by Cai et al. (2010), plays a central role in such algorithms. SVT is a well defined matrix arithmetic that can be viewed as the combination of the *singular value decomposition* (SVD) and *soft thresholding*. Yet different matrix completion algorithms vary in formula more or less, they rely on SVT to optimize the objective iteratively. More specifically, we concentrate on the path of Ma et al. (2011) and Mazumder et al. (2010).

In addition, because of the pervasive use of SVT, SVD becomes the chief heavy-lifting in computation when dealing with large scale problems. In existing literatures, several types of truncated SVD implementations are exploited for acceleration. As a summary, Cai et al. (2010) and Mazumder et al. (2010) apply the state-of-the-art truncated SVD package `PROPACK` (Larsen, 1998); Ma et al. (2011) embed a linear-time randomized SVD implementation in Drineas et al. (2006).

1.3 Organization

Our work is primarily motivated by Ma et al. (2011) and Mazumder et al. (2010). Although they are independently derived from different angles, they actually share the same framework. Following the name in Ma et al. (2011), we call the scheme *fixed point iteration* algorithm.

In the thesis, we derive a generalized form of convergence rate of the fixed point iteration algorithm, and renew some proofs with a more concise version. Based on the discussion of results, we propose a much more efficient algorithm. The new algorithm changes the step size adaptively with approximation. Numerical results support the approximation strategy, as a consequence, it effectively enhances the speed and remains all the good properties of the ascent algorithms.

The later chapters cover the above content and are organized as follows.

Chapter 2 illustrates the motivation of low-rank matrix completion and the non-expansive property of SVT. It also clarifies the relationship between low-rank matrix completion and other models.

Chapter 3 characterizes the framework of fixed point iteration algorithm, and derives the convergence property as well as the rate. From the convergence results, a new algorithm is proposed. Besides, implementation related issues are discussed.

Chapter 4 records the consequences of numerical experiments. It is clear that our proposed algorithm achieves the same precision with baseline methods in considerably fewer loops. Meanwhile, it benefits from the truncated SVD acceleration as well.

Chapter 5 concludes the whole thesis and figures out the potential directions for future research.

Chapter 2

Low-rank Matrix Completion via Nuclear Norm Minimization

2.1 Motivation

Data is commonly stored and presented in matrix form. In some circumstances, the data matrix itself is of interest but with serious incompleteness, such as images, genes, user-item ratings etc. How to handle the missing values in data matrix?

Generally speaking, it is impossible to recover the repealed entries of an arbitrary matrix. However, the recovery becomes possible if the underlying matrix has low rank. Intuitively, we illustrate the statement by counting the number of free parameters in a matrix, or equivalently, the degrees of freedom.

For a matrix, the degrees of freedom can be counted from its SVD form. Given a matrix $X \in \mathbb{R}^{m\times n}$ of rank r, we can write the SVD as

$$X = U\Sigma V^T, \tag{2.1}$$

where both $U \in \mathbb{R}^{m\times r}, V \in \mathbb{R}^{n\times r}$ are orthonormal matrices, i.e., $U^TU = V^TV = I$, $\Sigma \in \mathbb{R}^{r\times r}$ is a square, diagonal matrix, whose diagonal elements $\sigma_1, ...\sigma_r$ are positive and in descent order. Therefore, the equivalent formula is

$$X = \sum_{i=1}^{r} \sigma_i u_i v_i^T, \tag{2.2}$$

where u_i, v_i are the column vectors of U, V respectively.

There are $r(m+n+1)$ parameters in total from the SVD form, as well as $r(r+1)$ constrains to be subtracted. The latter equals the number of upper (lower) triangular entries of U^TU and V^TV due to the orthonormality and symmetry.

Hence, the degrees of freedom of a matrix is $r(m+n-r)$, depending on the rank. For $r \ll \min\{m,n\}$ cases, the matrices are actually dominated by a lot fewer parameters than it looks like. Therefore, low-rank assumption is very helpful in the context of matrix completion.

More specifically, the theoretical meaningfulness of nuclear norm heuristic in low-rank matrix completion is validated by several recent studies under suitable settings. For example, the following is a typical result from Candès and Recht (2009).

Theorem 2.1. *Suppose $M \in \mathbb{R}^{m\times n}$ is of rank r, denote by $M = \sum_{i=1}^{r} \sigma_i u_i v_i^T$ its SVD, where $\{u_i\}, \{v_i\}$ are sampled uniformly at random from all families of r orthonormal vectors independent from each other. Let $n_0 = \max(m,n)$, if we observe k entries of M with locations sampled uniformly at random, then there exist constants C and c such that if*

$$k \geq C n_0^{5/4} r \log n_0, \tag{2.3}$$

the minimizer of (1.2) is unique and equal to M with probability at least $1 - c n_0^{-3} \log n_0$. In addition, if $r \leq n_0^{1/5}$, then the same probability still holds for exact recovery given

$$k \geq C n_0^{6/5} r \log n_0. \tag{2.4}$$

Furthermore, (2.4) holds if the marginal distributions of $\{u_i\}$ and $\{v_i\}$ are uniform, regardless of the dependency of $\{u_i\}$ and $\{v_i\}$.

Under the settings, Theorem 2.1 characterizes the minimal number of entries for both generic and low-rank matrix reconstruction. With high probability, we need $O(n_0^{5/4}\log n_0)$ entries to recover an arbitrary matrix as (2.3); however, when the matrix is a low-rank one, the required entries reduce to $O(n_0^{6/5}\log n_0)$ as (2.4). Moreover, the unique solution is given by a convex optimization program (1.2), which is typically more computationally tractable.

2.2 Problem Formulation

For low-rank matrix completion, the common formulation of nuclear norm heuristic is

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \|X\|_* \\ \text{subject to} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M), \end{aligned} \tag{2.5}$$

where $\|X\|_* = \sum \sigma_i$ is the nuclear norm, M is the observed matrix, Ω is the corresponding entries, and $\mathcal{P}_\Omega(X)$ is defined as

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

However, solving (2.5) strictly can be abrupt, since it does not account for potential noise in observations. Even if the sample is indeed noiseless, as the objective of (2.5) is not a simple function (e.g., linear programming), the computation procedure is not able to follow the equality constraint either. The first step is to relax the equality constraint to inequality,

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \|X\|_* \\ \text{subject to} \quad & \|\mathcal{P}_\Omega(X - M)\|_F \le \delta, \end{aligned}$$

where $\delta > 0$ and $\|X\|_F = \sqrt{\sum X_{ij}^2}$ is the Frobenius norm.

More specifically, the corresponding *Lagrange* multiplier form is more appreciated, as it makes an unconstrained optimization problem,

$$\underset{X}{\text{minimize}} \quad \mu\|X\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(X-M)\|_F^2. \tag{2.6}$$

Here, $\mu > 0$. In order to solve (2.5), we should set μ very close to 0, and this returns the solution with sufficiently small error. For general purpose, we set the tolerance to different noise levels by adjusting μ.

It is worth noting that (2.5) and its relaxation (2.6) are not the unique choice, for example, another influential formulation adds penalization of the squared Frobenius norm of X (Cai et al., 2010).

$$\begin{aligned} \underset{X}{\text{minimize}} \quad & \mu\|X\|_* + \frac{1}{2}\|X\|_F^2 \\ \text{,subject to} \quad & \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned} \tag{2.7}$$

Instead of relaxing the equality constraint of (2.5), the strategy is to bound the sum of squares of the unobserved entries. The logic of (2.7) is perhaps not as intuitive as (2.6), however, it benefits from

2.3 SVT as Minimizer

Both (2.6) and (2.7) include a squared Frobenius norm related to X. Beyond the purpose of relaxation or penalization, the new formulation also eases the optimization. Originally, the nuclear norm itself is a non-smooth function, and its subdifferential does not have a clear form, which makes it hard to implement. However, when combined with a related squared Frobenius norm, the minimizer of the new objective is given by SVT.

SVT has a simple form and can be computed directly based on SVD. Using the SVD form of X in (2.2), then for any $\lambda > 0$, the SVT is defined as

$$S_\lambda(X) := \sum_{i=1}^{r} \max\{\sigma_i - \lambda, 0\} u_i v_i^T.$$

From the definition, SVT actually conducts soft thresholding on the singular values of the objective matrix. Hopefully, if X consists of many small singular values, SVT should return a low-rank matrix because of the shrinkage effect.

Lemma 2.2. *For any $\lambda > 0$ and $Y \in \mathbb{R}^{m\times n}$, we have*

$$S_\lambda(Y) = \operatorname*{argmin}_X \left(\lambda\|X\|_* + \frac{1}{2}\|X-Y\|_F^2\right).$$

We omit the proof that can be found in either Cai et al. (2010) or Mazumder et al. (2010). The former proof uses the subdifferential of nuclear norm, i.e.,

$$\partial\|X\|_* := \{UV^T + W : U^T W = \mathbf{0}, WV = \mathbf{0}, \|W\| \le 1\}, \tag{2.8}$$

where U, V are from the SVD of X, the spectral norm $\|W\|$ is the largest singular value of W, to verify the optimality condition (Lemma A.4); the latter proof conducts the minimizer by expanding the objective with their SVD forms.

Because of Lemma 2.2, SVT plays the key role in nuclear norm minimization procedure. The main advantage of SVT is its explicit form closely related to SVD, thus it is easy to implement. As a consequence, people always try to introduce some meaningful squared Frobenius norm term in order to exploit SVT, such as (2.6) and (2.7).

2.4 Non-expansiveness of SVT

As the shrinkage effect, SVT has non-expansive property in the Frobenius norm characterized by Lemma 2.3. The property is analogous to soft thresholding on vectors.

Lemma 2.3. *For any $\lambda > 0$, the SVT operator satisfies that*

$$\|S_\lambda(Y_1) - S_\lambda(Y_2)\|_F \le \|Y_1 - Y_2\|_F,$$

where the equality holds if and only if

$$Y_1 - Y_2 = S_\lambda(Y_1) - S_\lambda(Y_2).$$

The (equivalent) statement is seen in Ma et al. (2011) and Mazumder et al. (2010). The former proof is under different notations and derived based on the matrix inequalities; the latter proof makes use of the optimality condition as well as the result of Lemma B.2. Innovated by the latter approach, we provide a even simpler proof based on the optimality condition only.

Proof. Using Lemma 2.2 and Lemma (A.4), it satisfies that

$$\mathbf{0} \in \lambda\partial\|S_\lambda(Y_i)\|_* + S_\lambda(Y_i) - Y_i \quad i = 1, 2.$$

We specify the subgradients $g_i \in \partial\|S_\lambda(Y_i)\|_*$ satisfying the equalities, such that

$$Y_i = \lambda g_i + S_\lambda(Y_i) \quad i = 1, 2. \tag{2.9}$$

From the definition of subgradient, it follows that

$$\begin{cases} \mathrm{Tr}\left\{[S_\lambda(Y_2) - S_\lambda(Y_1)]^T g_1\right\} + \|S_\lambda(Y_1)\|_* \leq \|S_\lambda(Y_2)\|_* \\ \mathrm{Tr}\left\{[S_\lambda(Y_1) - S_\lambda(Y_2)]^T g_2\right\} + \|S_\lambda(Y_2)\|_* \leq \|S_\lambda(Y_1)\|_* \end{cases}$$

We sum these two inequalities up, and this yields that

$$\mathrm{Tr}\left\{[S_\lambda(Y_1) - S_\lambda(Y_2)]^T (g_1 - g_2)\right\} \geq 0.$$

Therefore, (2.9) simplifies to

$$\begin{aligned}
&\|Y_1 - Y_2\|_F^2 \\
=&\|S_\lambda(Y_1) - S_\lambda(Y_2) + \lambda(g_1 - g_2)\|_F^2 \\
=&\|S_\lambda(Y_1) - S_\lambda(Y_2)\|_F^2 + \lambda^2\|g_1 - g_2\|_F^2 \\
&\quad + 2\lambda \mathrm{Tr}\left\{\left[S_\lambda(Y_1) - S_\lambda(Y_2)\right]^T (g_1 - g_2)\right\} \\
\geq&\|S_\lambda(Y_1) - S_\lambda(Y_2)\|_F^2.
\end{aligned} \tag{2.10}$$

The last two terms of (2.10) are nonnegative. Hence, we complete the proof of first part.

From (2.10), it is obvious that the equality holds if and only if $g_1 = g_2$. From equation (2.9), this equals

$$Y_1 - Y_2 = S_\lambda(Y_1) - S_\lambda(Y_2),$$

which completes the proof. □

2.5 Relationship with the Lasso

Nuclear norm heuristic and the lasso (Tibshirani, 1996) are quite alike. Notice that nuclear norm is the ℓ_1 norm of the singular value vector,

$$\|X\|_* = \sum_{i=1}^{r} |\sigma_i| = \|\sigma\|_1,$$

thus (2.6) is equivalent to

$$\begin{aligned}
\underset{\sigma,U,V}{\text{minimize}} \quad & \mu\|\sigma\|_1 + \frac{1}{2}\left\|\left(\sum_{i=1}^{r}\sigma_i u_i v_i^T\right)_{ij} - M_{ij}\right\|_2^2 \\
\text{subject to} \quad & U^T U = I, \quad V^T V = I
\end{aligned} \tag{2.11}$$

where $(i, j) \in \Omega$.

We notice that (2.6) and (2.11) optimize σ, U, V jointly. However, if it can be divided into two steps, i.e., the first step returns the optimal U, V, then the second step is exactly a lasso problem to solve σ.

Similarly, one can also link rank minimization with best subset selection, as the rank is the ℓ_0 norm of singular value vector,

$$\operatorname{rank}(X) = \sum_{i=1}^{r} I(\sigma_i \neq 0) = \|\sigma\|_0.$$

Therefore, we can write the rank minimization problem as

$$\begin{array}{ll} \underset{\sigma,U,V}{\text{minimize}} & \mu\|\sigma\|_0 + \frac{1}{2}\left\| \left(\sum_{i=1}^{r} \sigma_i u_i v_i^T \right)_{ij} - M_{ij} \right\|_2^2 \\ \text{subject to} & U^T U = I, \quad V^T V = I \end{array}$$

By analogy, Mazumder et al. (2010) expect it should indicate a better recovery precision of nuclear norm heuristic over the rank minimization, since the lasso outperforms the best subset selection in many situations in terms of prediction accuracy with moderate sparsity settings (Hastie et al., 2009). Nevertheless, the guess may be not true as U, V subject to change, so that the comparison is not even on the same baseline.

2.6 Relationship with Matrix Factorization

Before nuclear norm heuristic was proposed for low-rank matrix completion, this kind of problems were (and are still) primarily handled via matrix factorization approach. In a nutshell, matrix factorization method fits the objective matrix as the product of two low-rank matrices, i.e. $M = AB^T$. The lower dimension of A, B is predetermined, such that they are explicitly low-rank matrices. Generally speaking, matrix factorization is not a convex program.

Interestingly, it can be shown that the solutions of the *maximum margin matrix factorization* (MMMF) (Srebro et al., 2004) and (2.6) coincide when A, B are assigned a dimension equal to the true rank of M or higher.

Briefly speaking, MMMF solves the problem

$$\underset{A,B}{\text{minimize}} \quad \frac{\mu}{2}\left(\|A\|_F^2 + \|B\|_F^2\right) + \frac{1}{2}\|\mathcal{P}_\Omega(AB^T - M)\|_F^2. \tag{2.12}$$

Equivalently, it could be reformulated as

$$\begin{aligned} \underset{X,A,B}{\text{minimize}} \quad & \frac{\mu}{2}\left(\|A\|_F^2 + \|B\|_F^2\right) + \frac{1}{2}\|\mathcal{P}_\Omega(X - M)\|_F^2 \\ \text{subject to} \quad & X = AB^T. \end{aligned} \tag{2.13}$$

This looks much like (2.6). To prove the equivalence, it suffices to show that $\|X\|_*$ is the solution of

$$\begin{aligned} \underset{A,B}{\text{minimize}} \quad & \frac{1}{2}\left(\|A\|_F^2 + \|B\|_F^2\right) \\ \text{subject to} \quad & X = AB^T. \end{aligned} \tag{2.14}$$

Recall the proof of Lemma B.2, it has been clarified that $\|X\|_*$ is the solution of the dual problem (B.2). For convenience, we sketch it here.

$$\begin{aligned} \underset{W_1,W_2}{\text{minimize}} \quad & \frac{1}{2}\left(\text{Tr}(W_1) + \text{Tr}(W_2)\right) \\ \text{subject to} \quad & \begin{bmatrix} W_1 & X \\ X^T & W_2 \end{bmatrix} \succeq 0. \end{aligned}$$

It is straightforward that for any matrix P, PP^T is positive semidefinite. Thus, let $P = \begin{pmatrix} A \\ B \end{pmatrix}$, then $PP^T = \begin{pmatrix} AA^T & X \\ X^T & BB^T \end{pmatrix} \succeq 0$ is eligible for (B.2) with $W_1 = AA^T$ and $W_2 = BB^T$, and the objective equals (2.13), i.e.,

$$\frac{1}{2}\left(\text{Tr}(W_1) + \text{Tr}(W_2)\right) = \frac{1}{2}\left(\|A\|_F^2 + \|B\|_F^2\right).$$

Therefore, it is clear that (2.13) is contained in (B.2).

We clarify that (2.13) could reach the minimum of (B.2), which is obtained at

$$(W_1 = U\Sigma U^T, W_2 = V\Sigma V^T).$$

Here, (U, V, Σ) is the SVD triplet of X. This is equivalent to

$$(A = U\Sigma^{1/2}, B = V\Sigma^{1/2}),$$

and it is eligible for (2.13). The form of the optimal (A, B) implies why we need their dimension to be at least the rank of M.

We illustrate the relationship between low-rank matrix completion and (maximum margin) matrix factorization. Although proven to be equivalent to (2.13) when the rank is properly specified, (2.6) handles the problem in a better way. The advantages of (2.6) over (2.13) include:

1. (2.6) is a convex optimization program, and hence free of the local minima issues that (2.13) might suffer from.
2. (2.6) learns the latent rank automatically, and hence avoids the subjectivity of "guessing" the rank when dealing with (2.13).
3. Even though a large dimension could be assigned to A, B in (2.13) to ensure the equivalence with (2.6), the computation cost of the former would inflate dramatically.

Chapter 3

Fixed Point Iteration Algorithm

3.1 Soft-Impute

We begin with a special case of fixed point iteration algorithm, which is the *soft-Impute* algorithm proposed by Mazumder et al. (2010).

Using the fact

$$X = \mathcal{P}_\Omega(X) + \mathcal{P}_{\Omega^c}(X),$$

where Ω^c is the complement of Ω, we can write (2.6) as

$$\underset{X}{\text{minimize}} \;\; \mu\|X\|_* + \frac{1}{2}\|X - \left(\mathcal{P}_{\Omega^c}(X) + \mathcal{P}_\Omega(M)\right)\|_F^2. \tag{3.1}$$

By Lemma 2.2, it satisfies that

$$X^* = S_\mu\left(\mathcal{P}_{\Omega^c}(X^*) + \mathcal{P}_\Omega(M)\right), \tag{3.2}$$

where X^* is a solution of (2.6).

From (3.2), X^* is easily recognized as a fixed point of (2.6). Therefore, the optimization can be converted into the iterative scheme with the recursive relation

$$X_{k+1} \leftarrow S_\mu \left(\mathcal{P}_{\Omega^c}(X_k) + \mathcal{P}_\Omega(M)\right).$$

The procedure is repeated until convergence, as the following algorithm.

Algorithm 1 Soft-Impute

```
1: Inputs:
       $\mathcal{P}_\Omega(M), \mu > 0$
2: Initialize:  $X_{curr} \leftarrow X_0$
3: while not converge do
4:     $X_{new} \leftarrow S_\mu \left(\mathcal{P}_{\Omega^c}(X_{curr}) + \mathcal{P}_\Omega(M)\right)$
5:     $X_{curr} \leftarrow X_{new}$
6: end while
7: $X^\mu \leftarrow X_{new}$
8: Outputs:  $X^\mu$
```

The key step 4 can be viewed as a two-step procedure:

1. Given the current estimate X_curr, adjust the entries in the observed subset Ω with observations.
2. Search the optimal matrix based on the polished current estimate.

3.2 Fixed Point Iteration

We introduce the fixed point iteration algorithm, which is actually a more generalized version of soft-Impute algorithm.

The strategy behind fixed point iteration dates back to Hale et al. (2007) for compressed sensing problem, before Ma et al. (2011) develop its matrix form. The idea is to construct an objective function, whose fixed point is a solution of (2.6). In this approach, solving

(2.6) is equivalent to searching the fixed point, which is completed by iteratively updating the fixed point equation until convergence.

Still denote by X^* a solution of (2.6), then from the optimality condition Lemma A.4, it satisfies that

$$\mathbf{0} \in \mu\partial\|X^*\|_* + \mathcal{P}_\Omega(X^* - M),$$

or equivalently,

$$\mathbf{0} \in \tau\left(\mu\partial\|X^*\|_* + \mathcal{P}_\Omega(X^* - M)\right) + (X^* - X^*),$$

with a constant $\tau > 0$. This can be rearranged as

$$\mathbf{0} \in \tau\mu\partial\|X^*\|_* + X^* - Y^*,$$

where $Y^* = X^* - \tau\mathcal{P}_\Omega(X^* - M)$.

Hence, from the optimality condition Lemma A.4, it suggests that

$$\begin{aligned} X^* &= \underset{X}{\operatorname{argmin}} \left(\tau\mu\|X\|_* + \frac{1}{2}\|X - Y^*\|_F^2\right) \\ &= S_{\tau\mu}\left(X^* - \tau\mathcal{P}_\Omega(X^* - M)\right), \end{aligned} \tag{3.3}$$

the second equation is from Theorem 2.2.

Under the construction, X^* is a fixed point again. The only difference is the change of update rule,

$$X_{k+1} \leftarrow S_{\tau\mu}\left(X_k - \tau\mathcal{P}_\Omega(X_k - M)\right).$$

Not surprisingly, it becomes exactly the soft-Impute algorithm if $\tau = 1$. We will show later in section 3.4 that τ is an important factor in the convergence rate. Moreover, our proposed algorithm gain its speed by making good use of τ.

Algorithm 2 Fixed Point Iteration

```
1: Inputs:
       $\mathcal{P}_\Omega(M), \mu > 0, \tau > 0$
2: Initialize:  $X_{curr} \leftarrow X_0$
3: while not converge do
4:     $X_{new} \leftarrow S_{\tau\mu}\left(X_{curr} - \tau\mathcal{P}_\Omega(X_{curr} - M)\right)$
5:     $X_{curr} \leftarrow X_{new}$
6: end while
7: $X^\mu \leftarrow X_{new}$
8: Outputs:  $X^\mu$
```

3.3 Convergence Property

We discuss the convergence property of fixed point iteration algorithm, which naturally includes that of the soft-Impute algorithm as well. A few similar results can be found in Ma et al. (2011) and Mazumder et al. (2010), however, we either generalize the conclusion or alternate in more concise ways.

For simplicity, we denote by $J_\tau(X)$ the matrix adjustment operation of step 4,

$$J_\tau(X) = X - \tau\mathcal{P}_\Omega(X - M), \tag{3.4}$$

such that (3.3) can be written as

$$X^* = S_{\tau\mu}\left(J_\tau(X^*)\right). \tag{3.5}$$

Lemma 3.1. *For $0 < \tau < 2$, $J_\tau(X)$ satisfies the inequality*

$$\|J_\tau(X_1) - J_\tau(X_2)\|_F \le \|X_1 - X_2\|_F,$$

Moreover, the equality holds if and only if $J_\tau(X_1) - J_\tau(X_2) = X_1 - X_2$.

Lemma 3.1 shows that $J_\tau(X)$ is non-expansive in the Frobenius norm given $0 < \tau < 2$. A similar result could be found in Ma et al. (2011), however, their proof is based on the spectral norm inequality, and the upper bound of τ is given in an implicit form. Compare with that, our proof is straightforward and the conclusion is explicit.

Proof. For $0 < \tau < 2$, it follows that

$$\begin{aligned}
&\|J_\tau(X_1) - J_\tau(X_2)\|_F^2 \\
=&\|(X_1 - X_2) - \tau\mathcal{P}_\Omega(X_1 - X_2)|_F^2 \\
=&\|X_1 - X_2\|_F^2 - 2\tau\mathrm{Tr}\left((X_1 - X_2)^T\mathcal{P}_\Omega(X_1 - X_2)\right) \\
&\quad + \tau^2\|\mathcal{P}_\Omega(X_1 - X_2)\|_F^2 \\
=&\|X_1 - X_2\|_F^2 + \tau(\tau - 2)\|\mathcal{P}_\Omega(X_1 - X_2)\|_F^2 \\
\leq&\|X_1 - X_2\|_F^2,
\end{aligned}$$

which proves the first statement.

Moreover, $\|J_\tau(X_1) - J_\tau(X_2)\|_F = \|X_1 - X_2\|_F$ is equivalent to $\mathcal{P}_\Omega(X_1) = \mathcal{P}_\Omega(X_2)$, from equation (3.4), it simplifies to

$$J_\tau(X_1) - J_\tau(X_2) = X_1 - X_2.$$

This completes the whole proof. $\square$

Combining Lemma 2.3 and 3.1 together, step 4 in Algorithm 2 is easily seen non-expansive in the Frobenius norm when $0 < \tau < 2$.

Proposition 3.2. *For any $0 < \tau < 2$, X_* is a solution of (2.6) if and only if*

$$\|S_{\tau\mu}\left(J_\tau(X_*)\right) - S_{\tau\mu}\left(J_\tau(X^*)\right)\|_F = \|X_* - X^*\|_F.$$

Proposition 3.2 is a necessary and sufficient condition for X_* to be a solution of (2.6). The proof is adapted from Ma et al. (2011).

Proof. The "only if" part is straightforward since X_* and X^* both satisfy (3.5).

In terms of the "if" part, from the fact, as well as Lemma 2.3 and 3.1, we have

$$\begin{aligned}
&\|X_* - X^*\|_F \\
=&\|S_{\tau\mu}\left(J_\tau(X_*)\right) - S_{\tau\mu}\left(J_\tau(X^*)\right)\|_F \\
\leq&\|J_\tau(X_*) - J_\tau(X^*)\|_F \\
\leq&\|X_* - X^*\|_F,
\end{aligned}$$

such that the equalities hold throughout, i.e.,

$$\|S_{\tau\mu}\left(J_\tau(X_*)\right) - S_{\tau\mu}\left(J_\tau(X^*)\right)\|_F = \|J_\tau(X_*) - J_\tau(X^*)\|_F = \|X_* - X^*\|_F,$$

which yields

$$S_{\tau\mu}\left(J_\tau(X_*)\right) - S_{\tau\mu}\left(J_\tau(X^*)\right) = J_\tau(X_*) - J_\tau(X^*) = X_* - X^*, \tag{3.6}$$

where the equalities are from Lemma 2.3 and 3.1 respectively.

Therefore, (3.5) suggests $S_{\tau\mu}\left(J_\tau(X_*)\right) = X_*$. Hence X_* satisfies (3.3) and is also a solution of (2.6). This completes the proof. □

With the lemmas and proposition, we claim the global convergence of Algorithm 2, i.e., the sequence generated by Algorithm 2 converges to a solution of (2.6). The proof is adapted from Ma et al. (2011).

Theorem 3.3. *For any $0 < \tau < 2$, the sequence $\{X_k\}$ obtained from Algorithm 2 converges to a solution of (2.6).*

Proof. From Lemma 2.3 and 3.1, for $0 < \tau < 2$, $\|X_k - X^*\|_F \geq 0$ is monotonically non-increasing,

$$\begin{aligned}&\|X_{k+1} - X^*\|_F \\ =&\|S_{\tau\mu}\left(J_\tau(X_k)\right) - S_{\tau\mu}\left(J_\tau(X^*)\right)\|_F \\ \leq&\|J_\tau(X_k) - J_\tau(X^*)\|_F \\ \leq&\|X_k - X^*\|_F,\end{aligned}$$

such that $\|X_k - X^*\|_F$ converges as $k \to \infty$,

$$\lim_{k\to\infty} \|X_k - X^*\|_F = \|X_* - X^*\|_F,$$

where X_* is a limit point of $\{X_k\}$.

We verify that X_* is a solution of (2.6) using Proposition 3.2. As $k \to \infty$, it satisfies that

$$\|X_{k+1} - X^*\|_F = \|S_{\tau\mu}\left(J_\tau(X_*)\right) - S_{\tau\mu}\left(J_\tau(X^*)\right)\|_F = \|X_k - X^*\|_F = \|X_* - X^*\|_F.$$

This completes the proof. □

Remark. Theorem 3.3 only claims that the fixed point iteration algorithms converge to one of solutions to (2.6). However, under the settings of Theorem 2.1, there is unique solution to (2.6) with high probability. As Theorem 2.1 is the theoretical foundation of matrix completion, it is reasonable to ignore the multi-solution issue (2.6).

Remark. For $0 < \tau < 2$, the equality condition of Lemma 2.3 and 3.1 enable us to eliminate the Frobenius norm and obtain equation (3.6). Therefore, any limit point of X_k is also a solution of (2.6).

As the boundary, $\tau = 2$ is a little special, as $\|J_2(X_1) - J_2(X_2)\|_F \equiv \|X_1 - X_2\|_F$ breaks the second statement of Lemma 3.1. Because of the non-expansive property of the SVT operator, $\|X_k - X^*\|_F$ still converges, since

$$\|X_{k+1} - X^*\|_F = \|S_{2\mu}\left(J_2(X_k)\right) - S_{2\mu}\left(J_2(X^*)\right)\|_F \leq \|J_2(X_k) - J_2(X^*)\|_F \equiv \|X_k - X^*\|_F.$$

However, X_k might end up jumping between two stationary points. Denote by X_∞^1 a limit point of X_k, from Lemma 2.3 it satisfies that

$$S_{2\mu}\left(J_2(X_\infty^1)\right) - S_{2\mu}\left(J_2(X^*)\right) = J_2(X_\infty^1) - J_2(X^*),$$

such that the subsequent one, denoting by X_∞^2, becomes

$$\begin{aligned} X_\infty^2 &= S_{2\mu}\left(J_2(X_\infty^1)\right) \\ &= S_{2\mu}\left(J_2(X^*)\right) + J_2(X_\infty^1) - J_2(X^*) \\ &= X^* + X_\infty^1 - X^* - 2\mathcal{P}_\Omega(X_\infty^1 - X^*) \\ &= X_\infty^1 - 2\mathcal{P}_\Omega(X_\infty^1 - X^*) \end{aligned} \tag{3.7}$$

Apply (3.7) once more on X_∞^2, and this yields $X_\infty^3 = X_\infty^1$, such that X_∞^1, X_∞^2 are stationary points if they are not equal.

Nevertheless, it should not be a problem to apply $\tau = 2$ in implementation, since the stationary situation can be easily quit by reducing τ when $\|X_k - X^*\|_F$ converges. Actually, our numerical experiments never witness a stationary situation. A possible illustration is that the stationary point either rarely exists or is too close to the solution to be discovered.

3.4 Convergence Rate

Theorem 3.3 guarantees the convergence of Algorithm 2 with $0 < \tau < 2$. Moreover, it is also desirable to learn the convergence speed. For the special case $\tau = 1$, Mazumder et al. (2010) conduct the convergence rate. Here we prove a more generalized result not limited to $\tau = 1$, yet it is stronger on $\tau = 1$. More significantly, it provides insights on the selection of τ, which leads to our proposed Algorithm 2.

Typically, it is more difficult to establish the convergence rate, and we need to rely on some additional conditions. Following the notations in Mazumder et al. (2010), we denote by $f_\mu(X)$ the objective of (2.6),

$$f_\mu(X) = \mu\|X\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(X - M)\|_F^2, \tag{3.8}$$

and define an intermediate variable $Q_{\tau,\mu}(X|\tilde{X})$ as

$$\begin{aligned} Q_{\tau,\mu}(X|\tilde{X}) = \mu\|X\|_* + \frac{1}{2}\|\mathcal{P}_\Omega(\tilde{X} - M)\|_F^2 + \frac{1}{2\tau}\|X - \tilde{X}\|_F^2 \\ + \operatorname{Tr}\left((X - \tilde{X})^T\mathcal{P}_\Omega(\tilde{X} - M)\right). \end{aligned} \tag{3.9}$$

$Q_{\tau,\mu}(X|\tilde{X})$ can be viewed as a second-order approximation of $f_\mu(X)$ around $\tilde{X}$, such that

$$Q_{\tau,\mu}(X_k|X_k) = f_\mu(X_k). \tag{3.10}$$

Moreover, it equals the objective of (3.3) divided by τ up to a constant. Hence, it satisfies that

$$X_{k+1} = \operatorname*{argmin}_X Q_{\tau,\mu}(X|X_k), \tag{3.11}$$

where $\{X^k\}$ is obtained from Algorithm 2.

As $f_\mu(X)$ is the objective we wish to minimize, it is desirable if the sequence $\{f_\mu(X_k)\}$ is non-increasing. A sufficient condition is

$$f_\mu(X_{k+1}) \leq Q_{\tau,\mu}(X_{k+1}|X_k), \tag{3.12}$$

which combined with (3.10), (3.12) leads to

$$f_\mu(X_{k+1}) \leq Q_{\tau,\mu}(X_{k+1}|X_k) \leq Q_{\tau,\mu}(X_k|X_k) = f_\mu(X_k). \tag{3.13}$$

For convenience, we temporarily take (3.12) for granted, and will come backward later in section 3.5. We are able to derive the convergence rate if the condition (3.12) holds throughout.

Lemma 3.4. *Suppose that $\bar{\tau}$ is carefully chosen such that (3.12) is fulfilled for all k, then for any X and $\{X_k\}$ generated by Algorithm 2, it obeys that*

$$\|X_{k+1} - X\|_F^2 - \|X_k - X\|_F^2 \leq 2\bar{\tau}\left(f_\mu(X) - f_\mu(X_{k+1})\right). \tag{3.14}$$

Proof. From (3.11), it is obvious that

$$f_\mu(X) - f_\mu(X_{k+1}) \geq f_\mu(X) - Q_{\bar{\tau},\mu}(X_{k+1}|X_k).$$

The convexity of $f_\mu(X)$ yields a lower bound of the first-order expansion around X_{k+1},

$$\begin{aligned} f_\mu(X) \geq \mu & \left\{\|X_{k+1}\|_* + \mathrm{Tr}\left(g^T(X - X_{k+1})\right)\right\} \\ & + \frac{1}{2}\|\mathcal{P}_\Omega(X_{k+1} - M)\|_F^2 + \mathrm{Tr}\left(\mathcal{P}_\Omega(X_{k+1} - M)^T(X - X_{k+1})\right), \end{aligned}$$

where $g \in \partial\|X_{k+1}\|_*$. Combine the above two inequalities, this yields

$$\begin{aligned} f_\mu(X) - f_\mu(X_{k+1}) \geq & -\frac{1}{2\bar{\tau}}\|X_{k+1} - X_k\|_F^2 \\ & + \mathrm{Tr}\left((X - X_{k+1})^T\left(\mu g + \mathcal{P}_\Omega(X_k - M)\right)\right). \end{aligned}$$

Since X_{k+1} minimizes $Q(X_{k+1}|X_k)$, it follows that

$$\mathbf{0} \in \mu\partial\|X_{k+1}\|_* + \mathcal{P}_\Omega(X_k - M) + \frac{1}{\bar{\tau}}(X_{k+1} - X_k).$$

Let g be the subgradient satisfying the equality, such that

$$\mu g + \mathcal{P}_\Omega(X_k - M) = -\frac{1}{\bar{\tau}}(X_{k+1} - X_k).$$

Plugging in this yields

$$\begin{aligned}
f_\mu(X) - f_\mu(X_{k+1}) &\geq -\frac{1}{2\bar{\tau}}\|X_{k+1} - X_k\|_F^2 - \frac{1}{\tau}\mathrm{Tr}\left((X - X_{k+1})^T(X_{k+1} - X_k)\right) \\
&= \frac{1}{2\bar{\tau}}\mathrm{Tr}\left((X_{k+1} - X_k)^T(X_{k+1} + X_k - 2X)\right) \\
&= \frac{1}{2\bar{\tau}}\mathrm{Tr}\left\{[(X_{k+1} - X) - (X_k - X)]^T[(X_{k+1} - X) + (X_k - X)]\right\} \\
&= \frac{1}{2\bar{\tau}}\left(\|X_{k+1} - X\|_F^2 - \|X_k - X\|_F^2\right),
\end{aligned}$$

which simplifies to (3.14) and completes the proof. □

Theorem 3.5. *Denote by X^* the solution of (2.6), then for any $k \geq 1$, the sequence $\{X_k\}$ generated by Algorithm 2 satisfies*

$$f_\mu(X_k) - f_\mu(X^*) \leq \frac{1}{2\bar{\tau}k}\|X^* - X_0\|_F^2. \tag{3.15}$$

if $\bar{\tau}$ satisfies condition (3.12) for all k.

Proof. Invoking Lemma 3.4, plug in $X = X_k$ and this yields

$$\|X_{k+1} - X_k\|_F^2 \leq 2\bar{\tau}\left(f_\mu(X_k) - f_\mu(X_{k+1})\right),$$

multiplying each inequality by its index and sum them together from $j = 0$ to k, we obtain

$$\begin{aligned}
\sum_{j=0}^{k} j\|X_{j+1} - X_j\|_F^2 \leq& 2\bar{\tau}\sum_{j=0}^{k} j\left(f_\mu(X_j) - f_\mu(X_{j+1})\right) \\
=& 2\bar{\tau}\left(\sum_{j=1}^{k+1} f_\mu(X_j) - (k+1)f_\mu(X_{k+1})\right).
\end{aligned} \tag{3.16}$$

Again for $X = X^*$, it follows that

$$\|X_{k+1} - X^*\|_F^2 - \|X_k - X^*\|_F^2 \leq 2\bar{\tau}\left(f_\mu(X^*) - f_\mu(X_{j+1})\right),$$

such that they sum to

$$\begin{aligned}\|X_{k+1}-X^*\|_F^2-\|X^*-X_0\|_F^2 =&\sum_{j=0}^{k}\left(\|X_{j+1}-X^*\|_F^2-\|X_j-X^*\|_F^2\right)\\ \leq& 2\bar{\tau}\sum_{j=0}^{k}\left(f_\mu(X^*)-f_\mu(X_{j+1})\right)\\ =&2\bar{\tau}\left((k+1)f_\mu(X^*)-\sum_{j=1}^{k+1}f_\mu(X_j)\right). \end{aligned} \tag{3.17}$$

Combine (3.16) and (3.17) together

$$\begin{aligned}2\bar{\tau}(k+1)\left(f_\mu(X_{k+1})-f_\mu(X^*)\right) \leq&\|X^*-X_0\|_F^2-\|X_{k+1}-X^*\|_F^2\\ \leq&\|X^*-X_0\|_F^2.\end{aligned} \tag{3.18}$$

Therefore, equivalently, for $k \geq 1$,

$$f_\mu(X_k)-f_\mu(X^*) \leq \frac{1}{2\bar{\tau}k}\|X^*-X_0\|_F^2.$$

□

As a direct comparison with the convergence rate of Mazumder et al. (2010), their result is

$$f_\mu(X_k)-f_\mu(X^*) \leq \frac{2}{k+1}\|X^*-X_0\|_F^2,$$

which is improved by our result if $\bar{\tau}=1$ is eligible,

$$f_\mu(X_k)-f_\mu(X^*) \leq \frac{1}{2k}\|X^*-X_0\|_F^2. \tag{3.19}$$

Remark. It is worth noting that $\bar{\tau}$ in Lemma 3.4 and Theorem 3.5 is not simply the τ assigned in Algorithm 2. Although, (3.19) is permanently true, or equivalently, $\bar{\tau}=1$ is always eligible.

3.5 Selection of τ

We discuss how to select τ properly for Algorithm 2. Based on the results, we depend on two criteria, i.e., Theorem 3.3 and 3.5. The theorems characterize two different styles of monotone convergence: $\|X_k - X^*\|_F$ for the former, and $f_\mu(X_k)$ for the latter.

We start from the aspect of Theorem 3.5. In section 3.4, the conduct of convergence rate of Algorithm 2 is based on condition (3.12), which guarantees the monotone convergence of $f_\mu(X_k)$ with the help of intermediate variable $Q_{\tau,\mu}(X_{k+1}|X_k)$. In order to verify the results, we check the condition (3.12), which is equivalent to

$$\begin{aligned}
& f_\mu(X_{k+1}) - Q_\mu(X_{k+1}|X_k) \\
=& \frac{1}{2}\left(\|\mathcal{P}_\Omega(X_{k+1}-M)\|_F^2 - \|\mathcal{P}_\Omega(X_k-M)\|_F^2\right) - \frac{1}{2\tau}\|X_{k+1}-X_k\|_F^2 \\
& - \operatorname{Tr}\left((X_{k+1}-X_k)^T\mathcal{P}_\Omega(X_k-M)\right) \\
=& \frac{1}{2}\operatorname{Tr}\left(\mathcal{P}_\Omega(X_{k+1}-X_k)^T\mathcal{P}_\Omega(X_{k+1}+X_k-2M)\right) - \frac{1}{2\tau}\|X_{k+1}-X_k\|_F^2 \\
& - \operatorname{Tr}\left((X_{k+1}-X_k)^T\mathcal{P}_\Omega(X_k-M)\right) \\
=& \frac{1}{2}\operatorname{Tr}\left((X_{k+1}-X_k)^T\mathcal{P}_\Omega(X_{k+1}+X_k-2M)\right) - \operatorname{Tr}\left((X_{k+1}-X_k)^T\mathcal{P}_\Omega(X_k-M)\right) \\
& - \frac{1}{2\tau}\|X_{k+1}-X_k\|_F^2 \\
=& \frac{1}{2}\operatorname{Tr}\left((X_{k+1}-X_k)^T\mathcal{P}_\Omega(X_{k+1}-X_k)\right) - \frac{1}{2\tau}\|X_{k+1}-X_k\|_F^2 \\
=& \frac{1}{2}\|\mathcal{P}_\Omega(X_{k+1}-X_k)\|_F^2 - \frac{1}{2\tau}\|X_{k+1}-X_k\|_F^2 \\
<& \, 0.
\end{aligned}$$

Hence, we obtain the range of τ for Lemma 3.4 and Theorem 3.5,

$$0 < \tau \leq \frac{\|X_{k+1}-X_k\|_F^2}{\|\mathcal{P}_\Omega(X_{k+1}-X_k)\|_F^2}, \tag{3.20}$$

where $\|\mathcal{P}_\Omega(X_{k+1} - X_k)\|_F^2 \neq 0$. Because Ω is a subset, it is obviously seen

$$\frac{\|X_{k+1} - X_k\|_F^2}{\|\mathcal{P}_\Omega(X_{k+1} - X_k)\|_F^2} \geq 1.$$

If $\|\mathcal{P}_\Omega(X_{k+1} - X_k)\|_F^2 = 0$, then any $\tau > 0$ is feasible. Therefore, $\tau \in (0, 1]$ is permanently eligible for (3.12), as well as Lemma 3.4 and Theorem 3.5. The discovery somehow illustrates the strategy of soft-Impute algorithm, as $\tau = 1$ is the maximum value of the absolutely feasible set.

Although, the upper bound result from Theorem 3.5 is just the worst case. This is caused by the proof procedure: in order to obtain a unified global upper bound, we require $\bar{\tau}$ to satisfy (3.20) for all values from 1 to k. That is to say, the $\bar{\tau}$ in the theorem has to be the minimum of the k eligible values. It is very likely that $\bar{\tau}$ is indeed close to or even reaches 1 when k grows.

However, if we remove this "redundant" constraint introduced for the purpose of global property proof, then τ does not necessarily satisfy all k constraints simultaneously. Furthermore, τ does not even have to be a constant through the iterations. As long as τ obeys the condition (3.20) in each step, then (3.13) demonstrates a monotone non-increase of $f_\mu(X_k)$, the core of Theorem 3.5.

Suppose we use the strategy that in each loop, $\tau > 0$ is judiciously chosen to satisfy the equality in (3.20). In this way, the real error $f_\mu(X_k) - f_\mu(X^*)$ is dramatically smaller than the theoretical upper bound, if sufficiently many τ's are significantly larger than the minimum τ. In fact, this is a common situation in low-rank matrix completion problem. Recall the setting and Theorem 2.1, Ω is usually a rather small subset, which implies that

$$\frac{\|X_{k+1} - X_k\|_F^2}{\|\mathcal{P}_\Omega(X_{k+1} - X_k)\|_F^2} \gg 1$$

should happen very often. From this viewpoint, choosing $\tau = 1$ is too conservative, as it strictly follows the strategy of Theorem 3.5, though it is optimal within that range.

We have seen that $\tau = 1$ is not a good choice. For Algorithm 2, the optimal constant value of τ should be 2. The first reason is that $\tau = 2$ could usually satisfy (3.20), and it approximately speeds up the convergence rate by two; the second reason is that from Theorem 3.3, $\tau = 2$ is the maximum to ensure the convergence of $\|X_k - X^*\|_F$. Thus, $\tau = 2$ is the best choice for Algorithm 2, if we restrict τ to be fixed.

3.6 New Algorithms

Fixing $\tau = 2$ still does not fully make use of Theorem 3.5. As we discuss in section 3.5, a better way is to update τ per loop under criterion (3.20). This idea naturally leads to a new fixed point iteration algorithm with line search, listed as Algorithm 3.

In Algorithm 3, $\tau = 2$ is set as the baseline option, and adjustment will take place whenever τ does not reach the bound provided by criterion (3.20).

Nevertheless, Algorithm 3 is just a prototype, because it has a serious drawback: the line search procedure is totally identical with a regular iteration of Algorithm 2, so is the computation complexity. The fact as well as our numerical results show that, for Algorithm 3, the reduction in number of loops and the line search cost usually cancel out or even worse.

Algorithm 3 Fixed Point Iteration with Line Search

```
1: Inputs:
       $\mathcal{P}_\Omega(M)$, $\mu > 0$, $0 < \gamma < 1$
2: Initialize:   $X_{curr} \leftarrow X_0$, $\tau \leftarrow 2$
3: while not converge do
4:     $X^{new} \leftarrow S_{\tau\mu}\left(X_{curr} - \tau\mathcal{P}_\Omega(X_{curr} - M)\right)$
5:     if $\tau < \frac{\|X_{new}-X_{curr}\|_F^2}{\|\mathcal{P}_\Omega(X_{new}-X_{curr})\|_F^2}$ then
6:         while $\tau < \frac{\|X_{new}-X_{curr}\|_F^2}{\|\mathcal{P}_\Omega(X_{new}-X_{curr})\|_F^2,}$ do
7:             $\tau \leftarrow \frac{\tau}{\gamma}$
8:             $X_{new} \leftarrow S_{\tau\mu}\left(X_{curr} - \tau\mathcal{P}_\Omega(X_{curr} - M)\right)$
9:         end while
10:    else if $\tau > \max\left(\frac{\|X_{new}-X_{curr}\|_F^2}{\|\mathcal{P}_\Omega(X_{new}-X_{curr})\|_F^2}, 2\right)$ then
11:        while $\tau > \max\left(\frac{\|X_{new}-X_{curr}\|_F^2}{\|\mathcal{P}_\Omega(X_{new}-X_{curr})\|_F^2}, 2\right)$ do
12:            $\tau \leftarrow \max(\gamma\tau, 2)$
13:            $X_{new} \leftarrow S_{\tau\mu}\left(X_{curr} - \tau\mathcal{P}_\Omega(X_{curr} - M)\right)$
14:        end while
15:    end if
16:    $X_{curr} \leftarrow X_{new}$
17: end while
18: $X^\mu \leftarrow X_{new}$
19: Outputs:   $X^\mu$
```

However, the strategy behind Algorithm 3 is still helpful. As the main time consuming module is the line search procedure, we come up with a simple method to get rid of the expensive line search. The idea is to approximate the current τ with the upper bound calculated from the last round. This new algorithm is summarized as Algorithm 4.

Algorithm 4 ADAPTIVE FIXED POINT ITERATION

```
1: Inputs:
       $\mathcal{P}_\Omega(M), \mu > 0$
2: Initialize:  $X_{curr} \leftarrow X_0, \tau \leftarrow 2$
3: while not converge do
4:     $X_{new} \leftarrow S_{\tau\mu}\left(X_{curr} - \tau\mathcal{P}_\Omega(X_{curr} - M)\right)$
5:     $\tau \leftarrow \max\left(\frac{\|X_{new}-X_{curr}\|_F^2}{\|\mathcal{P}_\Omega(X_{new}-X_{curr})\|_F^2}, 2\right)$
6:     $X_{curr} \leftarrow X_{new}$
7: end while
8: $X^\mu \leftarrow X_{new}$
9: Outputs:  $X^\mu$
```

3.7 Other Issues

Besides the technical part, we hereby consider a couple of details that are mainly about the implementation.

For algorithm 2, it takes a rather long time to converge if μ is too tiny. An intuitive explanation is that X_k almost remains stagnant, since the shrinkage size $\tau\mu$ is too small, see step 4. More formally, as $\tau\mu$ is very small, it approximately obeys

$$X_{k+1} = S_{\tau\mu}\left(X_k - \tau\mathcal{P}_\Omega(X_k - M)\right) \approx X_k - \tau\mathcal{P}_\Omega(X_k - M),$$

which implies the difference between X_{k+1} and X_k arises mostly in Ω. Recall the upper bound of τ by (3.20), this yields

$$\tau \leq \frac{\|X_{k+1} - X_k\|_F^2}{\|\mathcal{P}_\Omega(X_{k+1} - X_k)\|_F^2} \approx 1$$

Therefore, the tiny μ straits reaches the worst case (3.19) characterized by Theorem 3.5. Unlike other situation, τ is not that likely to vary dramatically over the loops, and the unique way from (3.19) to reduce error would be using a "warm start", i.e., initializing X_0 such that $\|X^* - X_0\|_F^2$ is small. In order to obtain a warm start, Ma et al. (2011) and Mazumder et al. (2010) suggest to solve problems with a series of decreasing μ until reach the destination. Usually, the first μ is sufficiently large and hence easy to handle. By construction, each current solution is a warm start for the subsequent problem. The strategy could be used to cross-validate μ as well. However, μ should not be too small, unless we are very confident that the observations are noiseless.

The next issue is how to accelerate SVD, since the SVT operator could be viewed as "SVD + softe thresholding", and the former is the heavy lifting. When data size grows rapidly, Exploiting the low-rank structure of X, we could reduce the full SVD to a truncated version concentrating only on the dominant singular values.

Essentially, there are two types of implementation, roughly speaking, deterministic and randomized versions. In terms of the former, the state-of-the-art software package `PROPACK` designed for computing the SVD of large and sparse matrices, is highly recommended under the circumstance. The performance of Ma et al. (2011) suggests that randomized version fits in with the low-rank matrix completion. This type of SVD implementations approximates the top-k singular value triplets with a random sample slightly larger than k of columns. Obviously, such algorithms enjoy dramatically lower computation and storage complexity, and the accuracy is provable in probability scheme (cf. Drineas et al., 2006; Halko et al., 2011).

Ideally, the truncated SVD should compute only those singular values $\sigma_i \geq \tau\mu$ and the corresponding singular vectors. However, for both versions, the dominant parameter is the number of singular values to compute. In practice, the parameter is selected through trial and error, e.g., Ma et al. (2011) increase the number by 1 if the non-expansiveness is violated 10 times, Mazumder et al. (2010) do the same adjustment when the last singular value is larger than $\tau\mu$.

The application of truncated SVD may significantly reduce the time cost in an iteration, yet it depends on the algorithm as well. Typically it makes additional operation than full SVD to calculate it the partial way, hence it outperforms the null only when the objective matrix is (approximately) severely low-rank. In the end, it does not influence the number of iterations to converge.

The last issue is the pro-proceeding of the solution. It is neither simple nor necessary to tune μ exactly to the best. Having obtained a solution, it is highly recommended to check the singular values and eliminate the redundancy if there exists. Generally speaking, too small singular values or too rapid decay in singular value sequence should be recognized as evidence of rank redundancy. We can simply drop the redundant triplets if the problem is not too serious, otherwise, we may have to try some new μ.

Chapter 4

Numerical Experiments

4.1 Settings and Main Results

The section summarizes the details of experiment settings, i.e., the implementation issues, the evaluation criteria, the platform etc. As a conclusion, the numerical experiments clarify two main results.

1. Our new algorithm 3, 4 perform the same as Algorithm 1, 2 in accuracy;
2. Algorithm 4 takes considerably fewer iterations than the original Algorithm 2.
3. Algorithm 4 is also able to exploit the problem structure to be accelerated by `PROPACK`, and it still outperforms Algorithm 1 in almost the same degree.

Because of result 1, we do not concentrate on the performance comparison with peer methods, circumstantial records could be found in the corresponding chapter or section of Ma et al. (2011) and Mazumder et al. (2010). More interestingly, the experiment results provide strong evidence upon result 2 and 3.

In experiment, the stopping rule is based on the fraction of X change. It is not only for the ease to calculate, but also because of the non-expansiveness of $\|X_k - X_{k-1}\|_F$. More specifically, the algorithms keep running until

$$\frac{\|X_{new} - X_{curr}\|_F}{\max\left(1, \|X_{curr}\|_F\right)} \leq 1 \times 10^{-4}.$$

As a summary, we compare the performance of the following methods.

1. *FPI1*. Algorithm 1 or equivalently Algorithm 2 with $\tau = 1$, which is the Soft-Impute algorithm in Mazumder et al. (2010).
2. *FPI2*. Algorithm 2 with $\tau = 2$, which is generally the fastest version of the FPC algorithm in Ma et al. (2011).
3. *FPILS*. Algorithm 3, which is the naïve algorithm with line search.
4. *AFPI*. Algorithm 4, which replaces the line search procedure of FPILS by simple approximation.

All the experiments are conducted on the platform with Intel Core i5-2450M, 2.50GHz CPU and 4GB RAM, and the MATLAB version is R2013b.

4.2 Simulation Study

In this simulation study, we generate the underlying low-rank matrix $M \in \mathbb{R}^{m\times n}$ of rank r with

$$M \leftarrow AB^T,$$

where $A \in \mathbb{R}^{m\times r}, B \in \mathbb{R}^{n\times r}$, and the entries of A, B are independent and identically distributed (i.i.d) samples from standard Gaussian distribution $N(0,1)$. It is easily seen that the entries of M have mean 0 and variance r.

In some studies, we need to handle observations with noise. This is fulfilled by introducing a noise matrix N. The elements of N are i.i.d. samples from Gaussian distribution with mean 0. For the purpose of comparability, we use the *signal-to-noise ratio* (SNR) to measure the noise level. SNR denotes the ratio of standard deviation between entries of M and N.

$$\text{SNR} := \frac{\sigma_M}{\sigma_N} = \frac{\sqrt{r}}{\sigma_N}$$

Hence, the distribution of N's entries is $N(0, r/\text{SNR}^2)$. Having obtained the matrix M or the noisy version $M+N$, the observed subset Ω is then uniformly sampled at random. We denote by SR the sampling ratio,

$$\text{SR} := \frac{|\Omega|}{mn}.$$

When evaluating the matrix completion consequence, we calculate two kinds of error, i.e., training error and test error, defined as

$$\text{training error} := \frac{\|\mathcal{P}_\Omega\left(X-(M+N)\right)\|_F^2}{\|\mathcal{P}_\Omega(M+N)\|_F^2}$$

and

$$\text{test error} := \frac{\|\mathcal{P}_{\Omega^c}\left(X-M\right)\|_F^2}{\|\mathcal{P}_{\Omega^c}(M)\|_F^2}.$$

The former measures the goodness-of-fit on observed set, and the latter calculates the prediction error with the underlying matrix on unobserved part. Both types of errors are standardized.

Without loss of generality, all the experiments are conduct on square matrices. The parameters of matrices are listed in Table 4.1. All the algorithms are tested on matrices with three different levels of noise, i.e., noiseless, SNR $=6$ and SNR $=9$. Throughout the study, we initialize with $X_0 = \mathcal{P}_\Omega(M+N)$ and use traditional full SVD to compute the SVT. Running time, recovered rank and the errors are averaged over 50 simulations. The results are summarized in Table 4.2 – 4.4.

From the results, we discover that all the algorithms perform almost the same in recovered rank and errors. The consequence is totally within expectation for FPI1, FPI2 and FPILS, because the problem is convex and the convergence of the above algorithms are guaranteed by Theorem 3.3 and 3.5. To our interest, the identical performance of AFPI with the above ones highly support our simplified τ-estimation method.

Table 4.1: Parameters of randomly generated matrices

dimension	rank	SR
100	10	0.5
200	10	0.4
500	20	0.25
1000	50	0.25

Under all these circumstances, the FPI algorithms complete matrices with test errors less than 0.1. They perform very well for noiseless (Table 4.2) case and high SNR (Table 4.3) case. Nevertheless, for difficult problems with low sampling ratio, high dimension and rank, we observe that the recovered rank inflates as the noise level increases (Table 4.4, $m = 500, 1000$).

In the aspect of running time, for rather easy problems with high sampling ratio, low dimension and true rank, FPI1 is usually the fastest one ($m = 100$). When dimension is larger than 100, FPI1 becomes the slowest one, as we expect. FPILS is almost as slow as FPI1, since the line search procedure is identical to a regular iteration, it fails to gain efficiency through line search. In most cases, FPI2 spends around 55% the running time of FPI1, the fact somehow supports our guess that (3.20) is always far greater than 1. Our proposed algorithm AFPI performs the best in difficult problems ($m = 500, 1000$), it could even cut down about 33% the running time from FPI2.

As we use the default full SVD in the tests, due to the homogeneity of core operation, the running time reflects the number of iterations (and line search procedures for FPILS). To understand how these algorithms work, we pick up the $m = 1000$ case of Table 4.3 to display the procedure perspectively. The results are summarized in Table 4.5 and Figure 4.2, 4.1.

Table 4.5 displays the number of iterations[1] required by each algorithms. Consistent with the running time in Table 4.3, FPI1 takes 76 formal loops to converge, compared with 25 loops for FPILS. However, in order to achieve the minimum number of formal

[1]equivalent iterations include the line search procedures of FPILS.

Table 4.2: Numerical results for randomly generated matrix completion with noise free observations, where $\gamma = \frac{2}{3}$, $\mu = \sqrt{m}$

	algorithm	time	rank	training error	test error
$m = 100$	FPI1	0.185	10	0.0311	0.0627
$r = 10$	FPI2	0.231	10	0.0311	0.0627
SR = 50%	FPILS	0.238	10	0.0311	0.0627
	AFPI	0.249	10	0.0311	0.0627
$m = 200$	FPI1	1.061	10	0.0348	0.0586
$r = 10$	FPI2	0.569	10	0.0348	0.0585
SR = 40%	FPILS	0.836	10	0.0348	0.0585
	AFPI	0.556	10	0.0348	0.0585
$m = 500$	FPI1	9.583	20	0.0373	0.0693
$r = 20$	FPI2	5.431	20	0.0373	0.0691
SR = 25%	FPILS	9.472	20	0.0373	0.0690
	AFPI	3.672	20	0.0373	0.0690
$m = 1000$	FPI1	114.414	50	0.0203	0.0460
$r = 50$	FPI2	62.797	50	0.0203	0.0458
SR = 25%	FPILS	110.594	50	0.0202	0.0457
	AFPI	41.917	50	0.0202	0.0458

Table 4.3: Numerical results for randomly generated matrix completion with noisy observations, where SNR $= 9$, $\gamma = \frac{2}{3}$; $\mu = \sqrt{m}$ for $m = 100, 200, 500$; $\mu = 1.5\sqrt{m}$ for $m = 1000$

	algorithm	time	rank	training error	test error
$m = 100$	FPI1	0.154	10	0.0392	0.0662
$r = 10$	FPI2	0.214	10	0.0391	0.0662
SR = 50%	FPILS	0.215	10	0.0391	0.0662
SNR = 9	AFPI	0.210	10	0.0391	0.0662
$m = 200$	FPI1	1.121	10	0.0437	0.0607
$r = 10$	FPI2	0.592	10	0.0437	0.0606
SR = 40%	FPILS	0.908	10	0.0437	0.0606
SNR = 9	AFPI	0.658	10	0.0437	0.0606
$m = 500$	FPI1	16.759	20	0.0455	0.0724
$r = 20$	FPI2	9.347	20	0.0455	0.0722
SR = 25%	FPILS	17.151	20	0.0454	0.0722
SNR = 9	AFPI	6.806	20	0.0454	0.0722
$m = 1000$	FPI1	84.250	50.43	0.0499	0.0920
$r = 50$	FPI2	46.672	50.29	0.0499	0.0919
SR = 25%	FPILS	83.443	50.19	0.0499	0.0918
SNR = 9	AFPI	30.115	50.24	0.0499	0.0918

Table 4.4: Numerical results for randomly generated matrix completion with noisy observations, where SNR $= 6$, $\gamma = \frac{2}{3}$; $\mu = \sqrt{m}$ for $m = 100, 200, 500$; $\mu = 1.5\sqrt{m}$ for $m = 1000$

	algorithm	time	rank	training error	test error
$m = 100$	FPI1	0.111	10	0.0491	0.0701
$r = 10$	FPI2	0.143	10	0.0491	0.0700
SR = 50%	FPILS	0.154	10	0.0491	0.0700
SNR = 6	AFPI	0.149	10	0.0491	0.0700
$m = 200$	FPI1	1.138	10	0.0549	0.0638
$r = 10$	FPI2	0.639	10	0.0549	0.0638
SR = 40%	FPILS	0.991	10	0.0549	0.0638
SNR = 6	AFPI	0.669	10	0.0549	0.0638
$m = 500$	FPI1	10.757	22.38	0.0551	0.0758
$r = 20$	FPI2	5.908	22.24	0.0551	0.0757
SR = 25%	FPILS	10.402	22.20	0.0551	0.0756
SNR = 6	AFPI	4.000	22.20	0.0551	0.0756
$m = 1000$	FPI1	98.823	78.70	0.0575	0.0995
$r = 50$	FPI2	54.647	78.19	0.0575	0.0993
SR = 25%	FPILS	98.183	78.00	0.0575	0.0991
SNR = 6	AFPI	36.120	78.00	0.0575	0.0991

Figure 4.1: Change of τ over formal iterations, where $m = 1000$, $r = 50$, SR = 25%, SNR = 9, $\mu = 1.5\sqrt{m}$

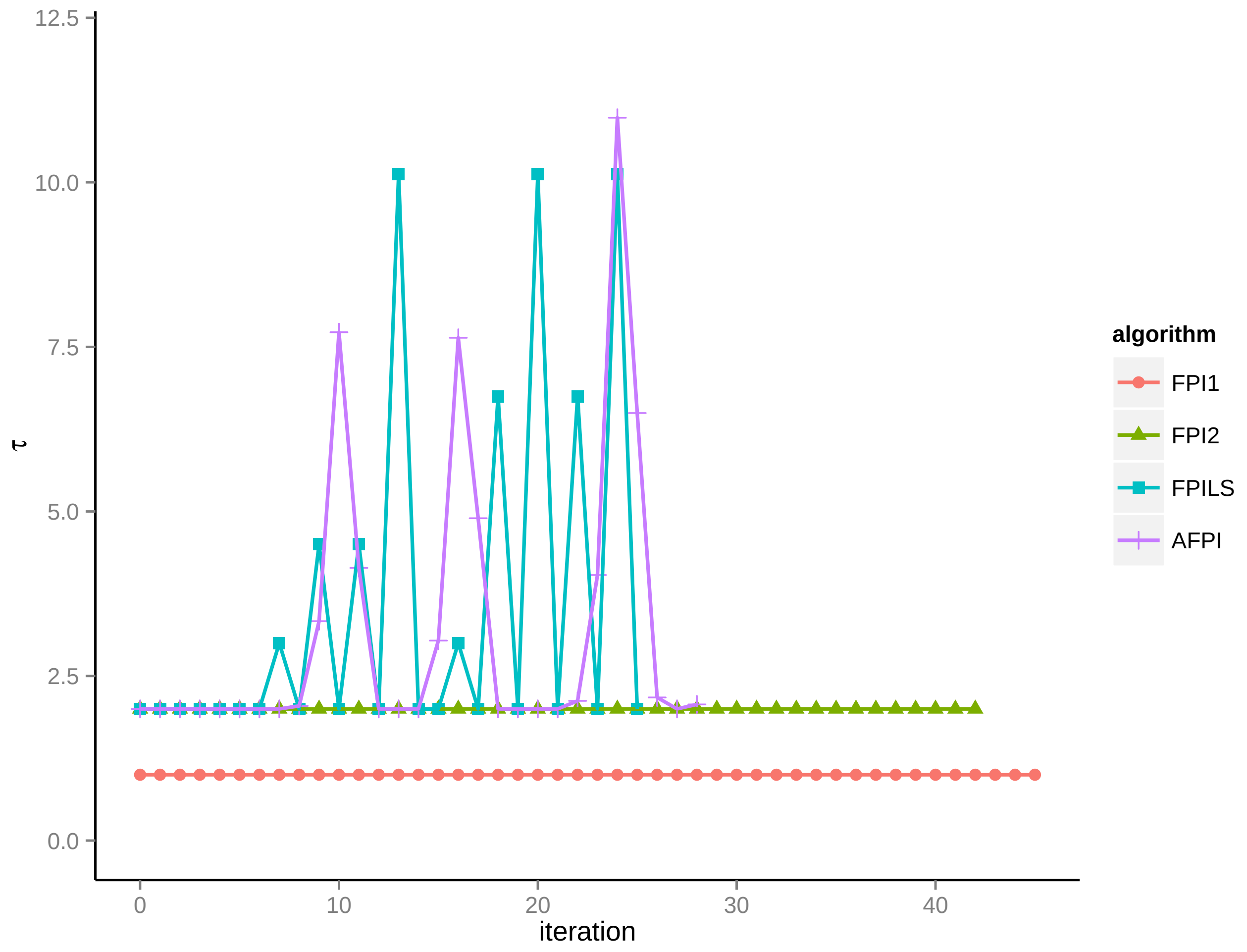


Figure 4.2: Change of $f_\mu(X)$ over formal iterations, where $m = 1000$, $r = 50$, SR $= 25\%$, SNR $= 9$, $\mu = 1.5\sqrt{m}$

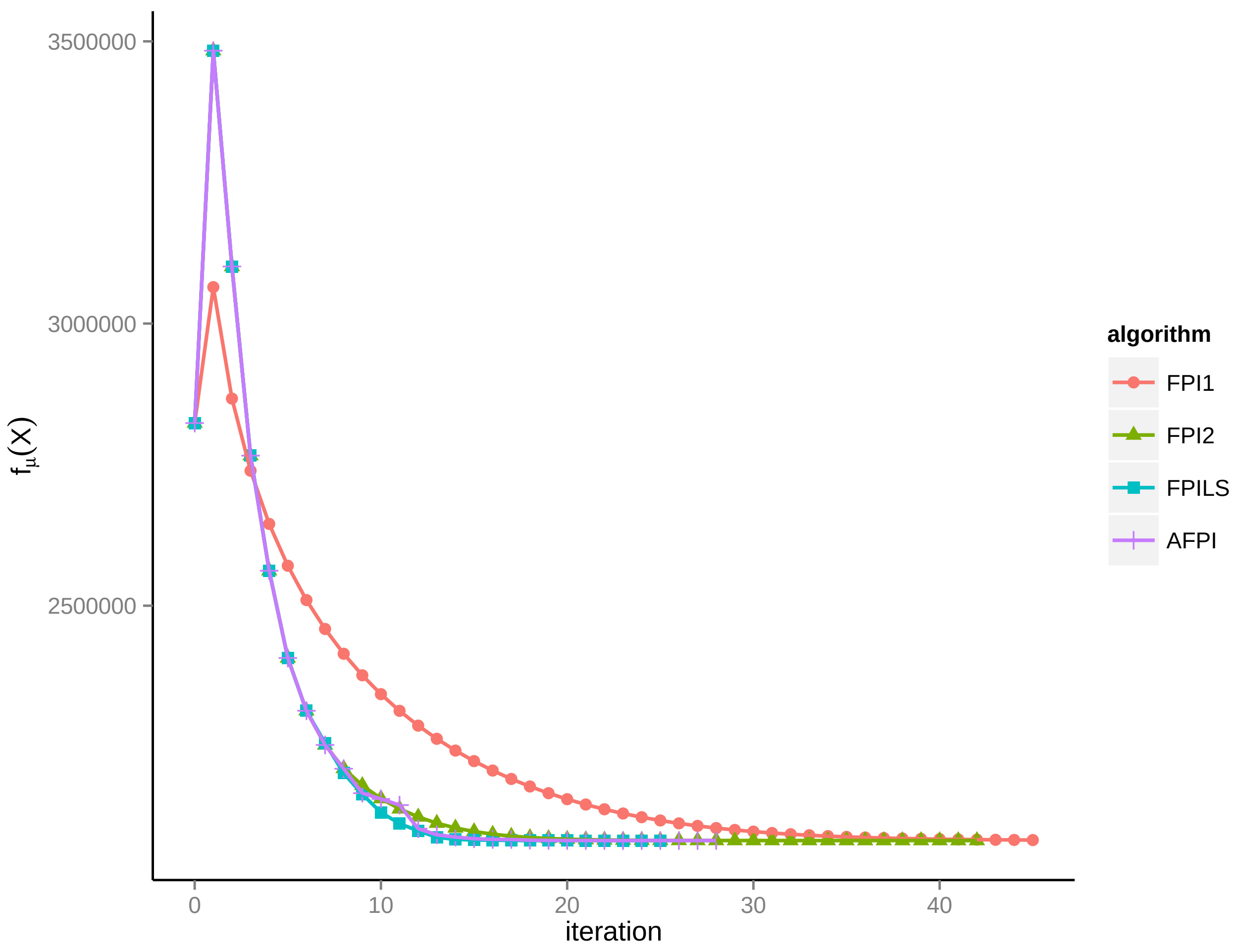


Figure 4.3: Part of Figure 4.2, highlight the difference between FPI2, FPILS and AFPI from iteration 10 to 20

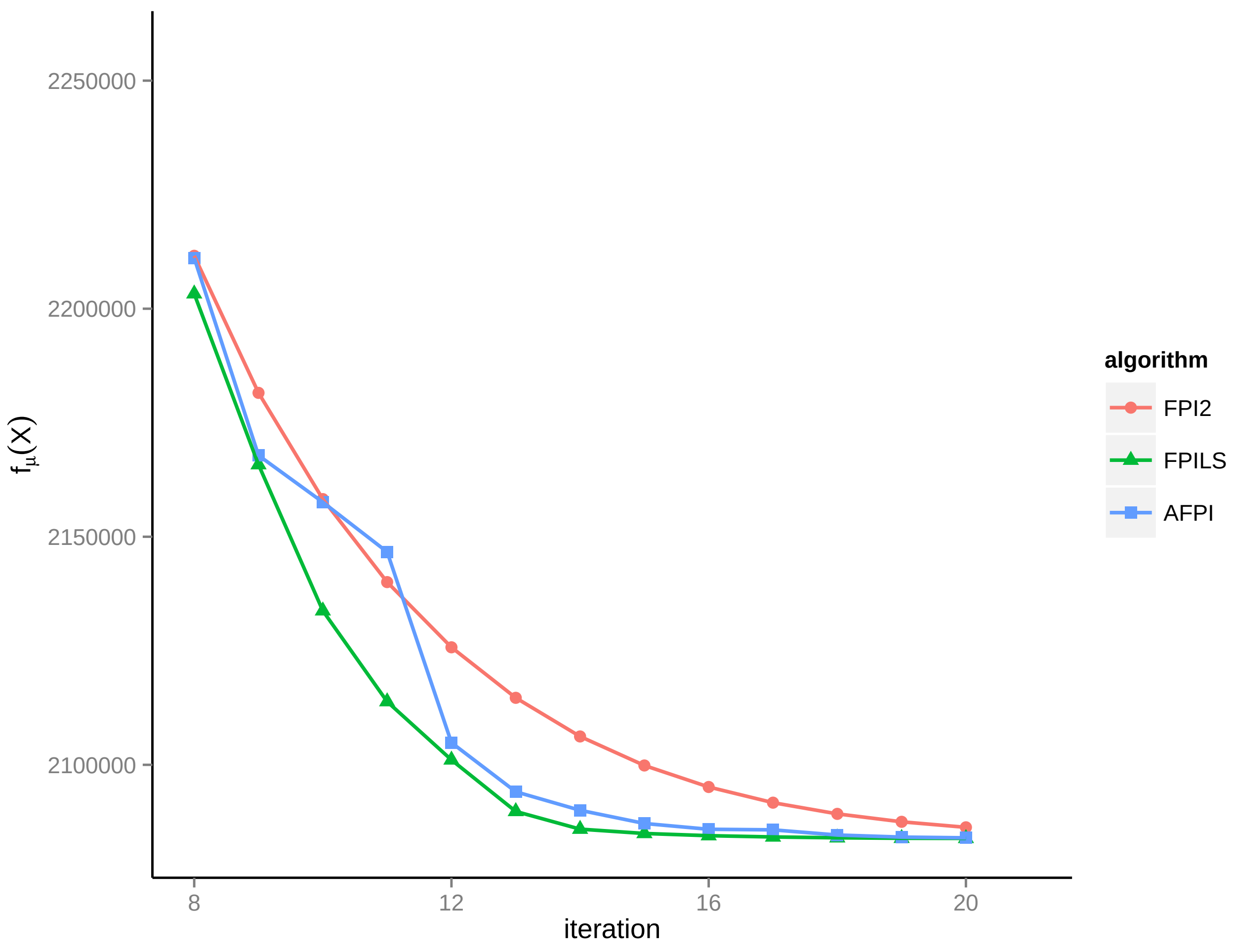


Table 4.5: Number of formal iterations and equivalent iterations, where $m = 1000$, $r = 50$, SR = 25%, SNR = 9, $\mu = 1.5\sqrt{m}$

algorithm	# formal iterations	# equivalent iterations
FPI1	76	76
FPI2	42	42
FPILS	25	73
AFPI	28	28

iterations, FPILS has to apply line search so frequently, that its equivalent iterations are almost as many as FPI1. However, AFPI eliminates the huge line search cost by replacing it with approximation without introducing heavy burden in computation.

Figure 4.1 illustrates the tendency of τ change. For most of the time, AFPI take values around $\tau = 2$; otherwise, their peaks come after one or several those of FPILS. Although this may cause a little delay (e.g., Figure 4.2, iteration = 10), the strategy always performs quite well. By setting $\tau = 2$ as the baseline, FPI2, FPILS, AFPI all enjoy much steeper descent than FPI1 at the beginning. However, when $f_\mu(X)$ starts to decrease flatly, for example, iteration ≥ 20, FPI2 becomes slower than FPILS and AFPI.

The difference in flat part is highlighted in Figure 4.3. FPILS goes down in the steepest path, and AFPI follows up the tendency in three iterations. Both algorithms are faster than FPI2. This is also reflected in the τ behaviors of FPI2 versus FPILS and AFPI in Figure 4.1.

As a conclusion, among the fixed point iteration algorithms above, AFPI is the optimal one in the sense that it obtains the same precision with the least time, as well as the fewest equivalent iterations on large matrices. Modified from FPILS, AFPI smartly updates τ without costly line search, and the simple strategy well approximates the consequence of line search with an ignorable cost.

4.3 Acceleration Result

For the completion of large matrices, full SVD is prohibitively expensive in computation. Compared with that, since only a small fraction of singular values are required, truncated SVD is far more time-economic under the circumstance. Since large-scale SVD is the most significant computational bottleneck, algorithms which well cooperate with truncated SVD always outperform the ones with better theoretical convergence rate but poorly fit to the purpose of applying truncated SVD. The numerical tests by Lin et al. (2010) show that `PROPACK` is usually slower than full SVD if computing more than 20% singular values. Therefore, in order to accelerate with it, desirable algorithms should frequently compute the SVD of approximately low-rank matrices.

For example, the convergence rate of soft-Impute/FPI1 algorithm is $O(\frac{1}{k})$, compared with $O(\frac{1}{k^2})$ of *accelerated Nestorov* algorithm by Ji and Ye (2009), which is the optimal convergence rate of first-order algorithms. Although, the SVT objective of the former has a special structure of "Low Rank + Sparse", i.e., the key step is

$$S_\mu \left(X_k + \mathcal{P}_\Omega(M - X_k) \right),$$

where X_k is a low-rank matrix and $\mathcal{P}_\Omega(M - X_k)$ is a sparse one. Mazumder et al. (2010) figure out that the structure is usually approximately low-rank, and hence enjoys a supreme bonus when implementing with `PROPACK`. Their numerical results present an obvious advantage in running time over the latter, when both embedded with `PROPACK`. The latter does not gain so much acceleration from truncated SVD as the former. Since the objective of the latter is a combination of X_k and X_{k-1}, it does not reduce to approximately low-rank quickly and stably, which is against the usage of truncated SVD.

The "Low Rank + Sparse" structure naturally holds for all the fixed point iteration algorithms above. For accelerated version, we still expect it to outperform FPI1. Because the truncated SVD does not influence the number of iterations to converge, and the operations with acceleration here are still homogeneous.

With the $m = 1000$ examples from Table 4.2 – 4.4, Table 4.6 presents the performance

Table 4.6: Running time of FPI1 and AFPI embedded with `PROPACK`. The examples are from the last rows of Table 4.2 – 4.4, where $m = 1000$, rank $= 50$, SR $= 25\%$, $\mu = \sqrt{m}$ for noiseless case; $\mu = 1.5\sqrt{m}$ for SNR $= 6$ and SNR $= 9$.

SNR	algorithm	time	reduction rate
noiseless	FPI1	69.969	38.85%
	AFPI	26.554	36.65%
9	FPI1	49.858	40.82%
	AFPI	17.977	40.30%
6	FPI1	67.245	31.95%
	AFPI	26.817	25.76%

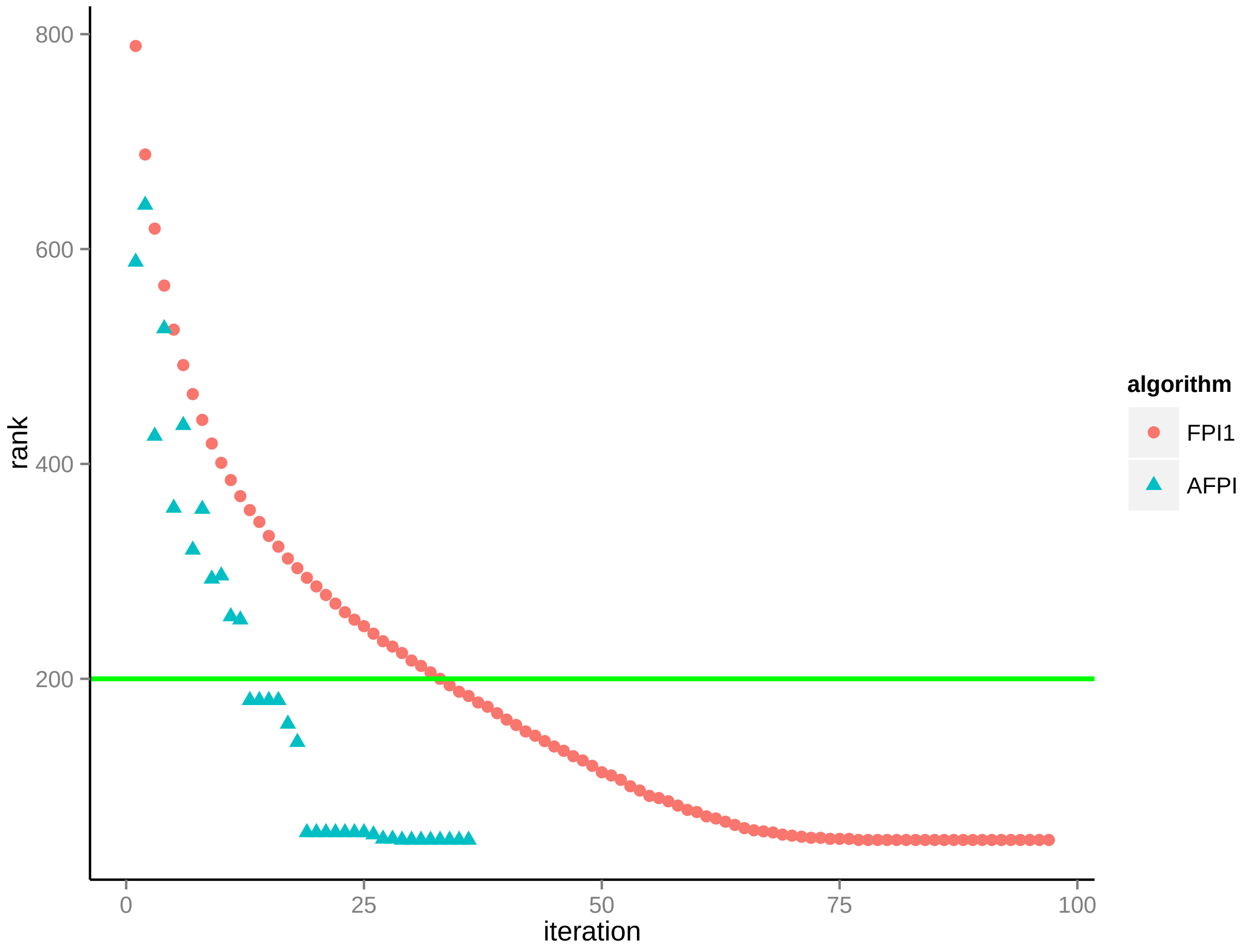
of accelerated FPI1 and AFPI. In terms of the acceleration, we substitute the full SVD with the truncated one, when the rank of current objective is smaller than $0.2m$. The results are averaged over 50 simulations, and the time reduction rate is calculated based on the running time of full SVD version in Table 4.2 – 4.4. Furthermore, we display the rank change of the noiseless example, as Figure 4.4.

From Table 4.6, by replacing the full SVD on (approximately) low-rank matrix with truncated SVD, the accelerated version saves a lot of time. The acceleration is even more significant and necessary as the dimension grows extremely high.

Figure 4.4 highlights the accelerated iterations explicitly, i.e., the points under the horizontal green line. The red curve displays the non-increasing trend of rank, such that truncated SVD keeps on as long as the rank of X_k decline to the green line. This illustrates how FPI1 exploits the structured SVD to accelerate. For AFPI, although the monotonicity slightly breaks at the beginning, the rank reaches the green line much faster than FPI1, and it descends monotonically after falling off the green line. The gaps of rank between neighbourhood iterations suggest the effect of adaptively changing τ.

Consequently, we observe that truncated SVD via `PROPACK` dramatically reduces the time cost of a single iteration, when computing sufficiently few singular value triplets. The fixed point iteration algorithms form the objective into "Low Rank + Sparse" structure, which is convenient to apply truncated SVD constantly. We have been aware that AFPI

Figure 4.4: Change of rank over iterations, where $m = 1000$, $r = 50$, SR $= 25\%$, $\mu = \sqrt{m}$ and noiseless



takes fewer steps to converge than any other peers from section 4.2, the results here further demonstrate that AFPI also takes advantages of acceleration earlier and better than the proven successful soft-Impute/FPI1 algorithm.

4.4 An Image Example

In order to verify the effectiveness of our algorithm on real data, as well as illustrate it in a visible approach, we present the algorithms on the image recovery. Images are stored in matrix forms, for example, a color image needs three matrices, representing red, green and blue, respectively. In this example, for simplicity, we consider the grayscale figure stored in one matrix. Each entry of the matrix stands for the grey scale of the corresponding pixel.

The dimension of the testing figure is 768×1024, and it is not an ideally (approximately) low-rank matrix consistent with the model assumption. Moreover, we do not apply truncated SVD either due to the above reason. The mission is to recover the matrix, of which 50% pixels are masked uniformly at random. In this example, fixed point iteration algorithms still succeed to recover good approximations, even though the assumptions are not fulfilled.

Table 4.7: Records of image recovery, where 50% pixels masked uniformly at random and the true rank is 768. Set $\mu = 1.25 \times \sqrt{1024}$, tol $= 1 \times 10^{-3}$

algorithm	time	rank	error ($\times 10^3$)
FPI1	118.81	335	4.46
FPI2	66.85	317	4.29
FPILS	82.85	315	4.25
AFPI	77.31	312	4.26

We notice that FPI2 is the fastest one, and the running time is still around 55% of FPI1. However, AFPI becomes much slower than before. This phenomenon can be illustrate by the stopping rule we choose. Because FPI1 and FPI2 strictly follows Theorem 3.3, $\|X_k - X_{k+1}\|_F$ decreases monotonically. AFPI, in contrast, make use of Theorem 3.5

Figure 4.5: Original Figure (768×1024), rank = 768

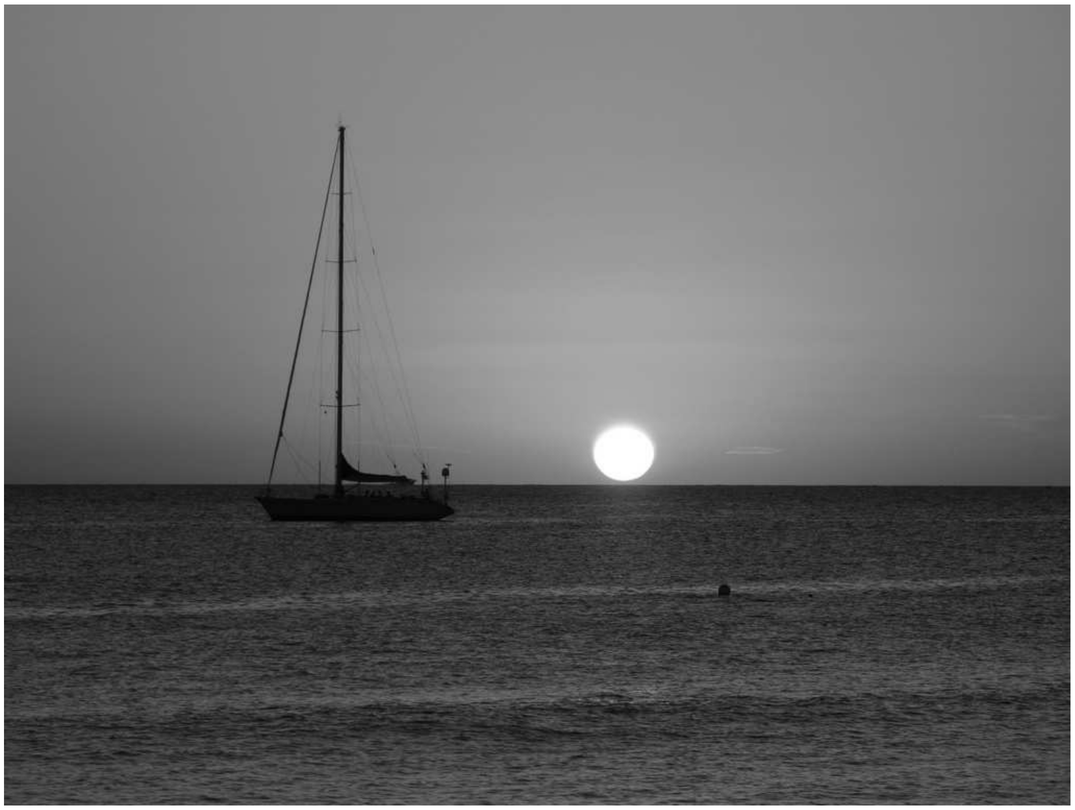


Figure 4.6: Remove 50% pixels of the original figure uniformly at random

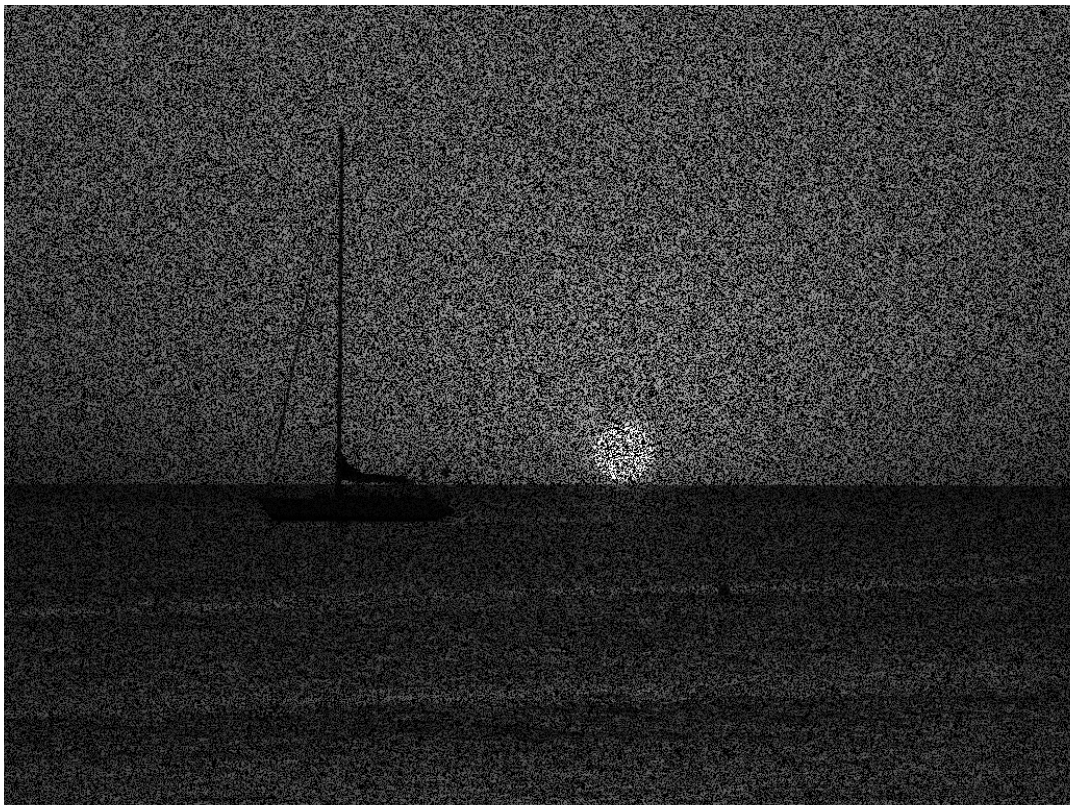


Figure 4.7: Figure recovered by FPI1, rank = 335, error = 4.46×10^3

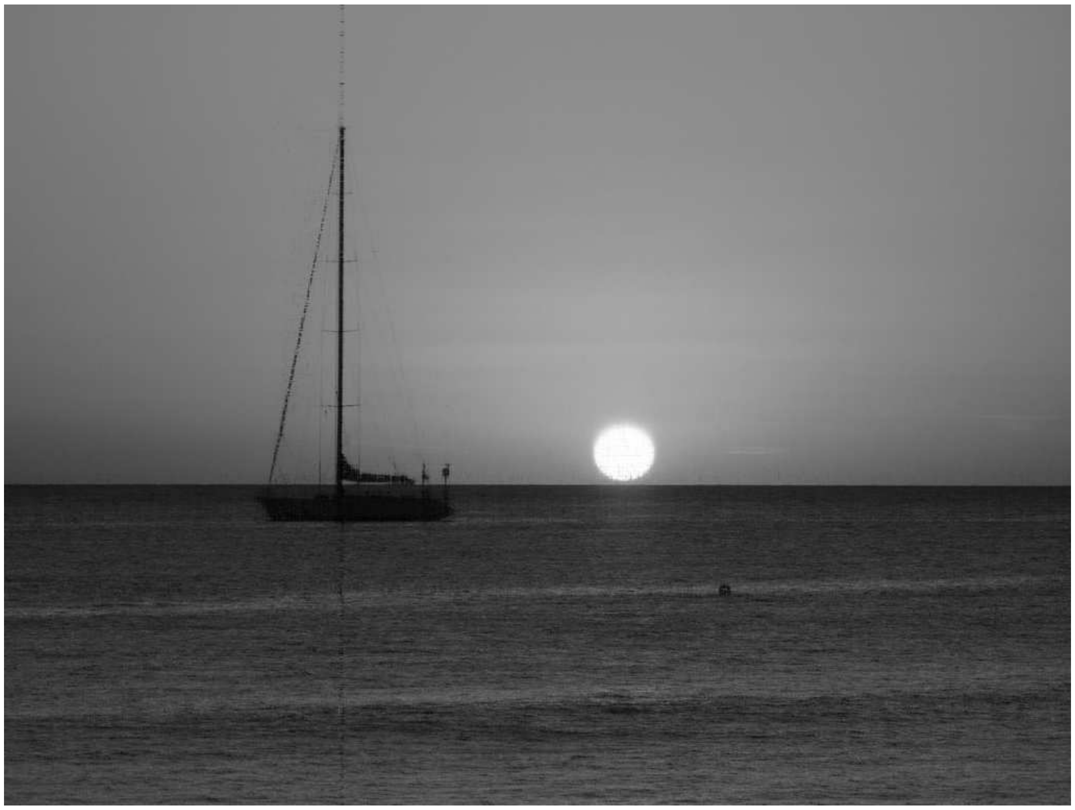


Figure 4.8: Figure recovered by FPI2, rank = 317, error = 4.29×10^3

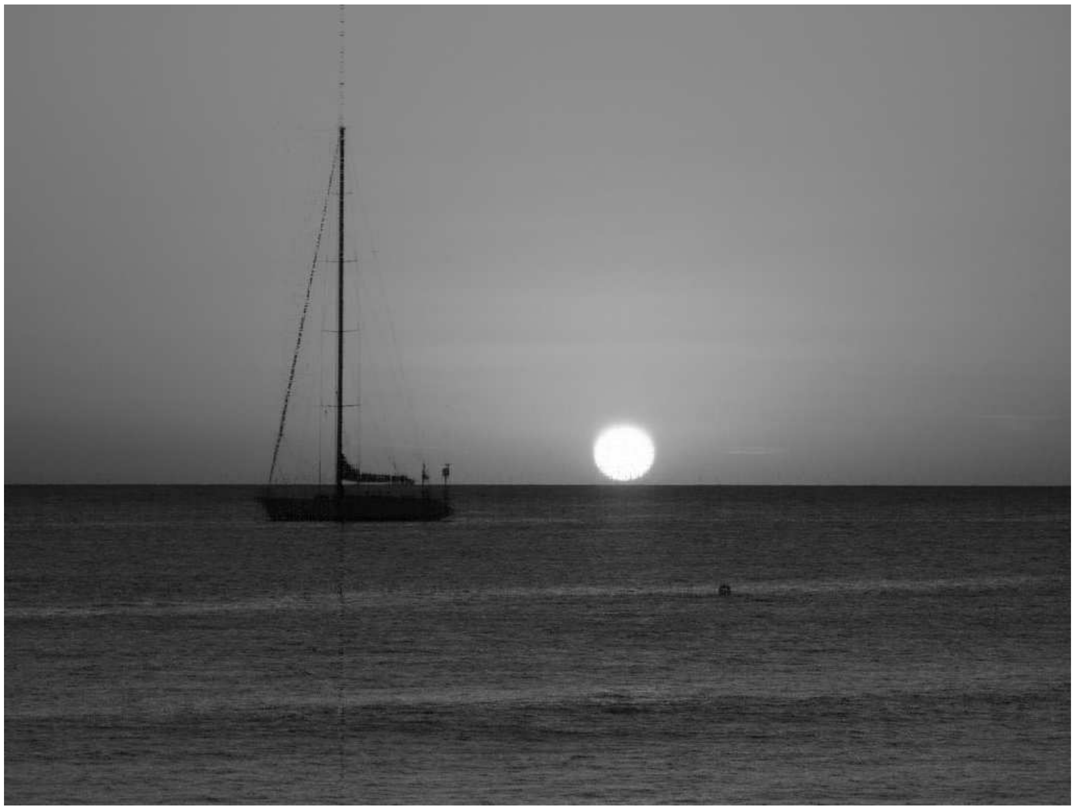


Figure 4.9: Figure recovered by FPILS, rank = 315, error = 4.25×10^3

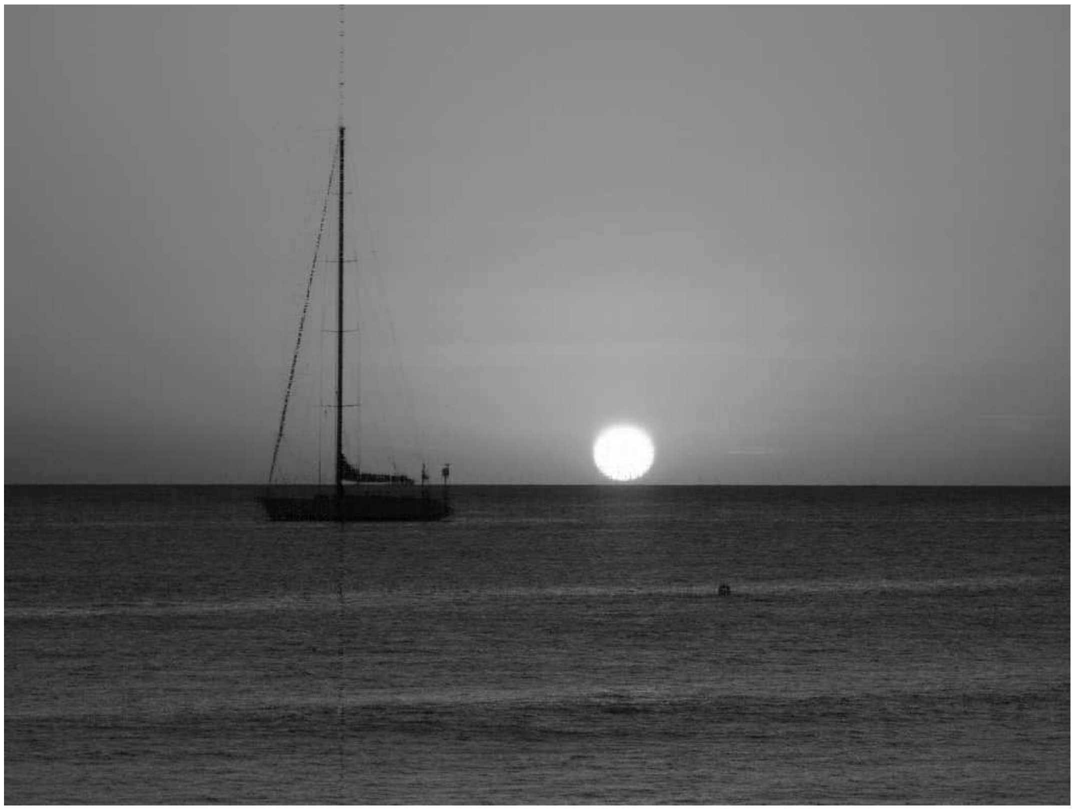
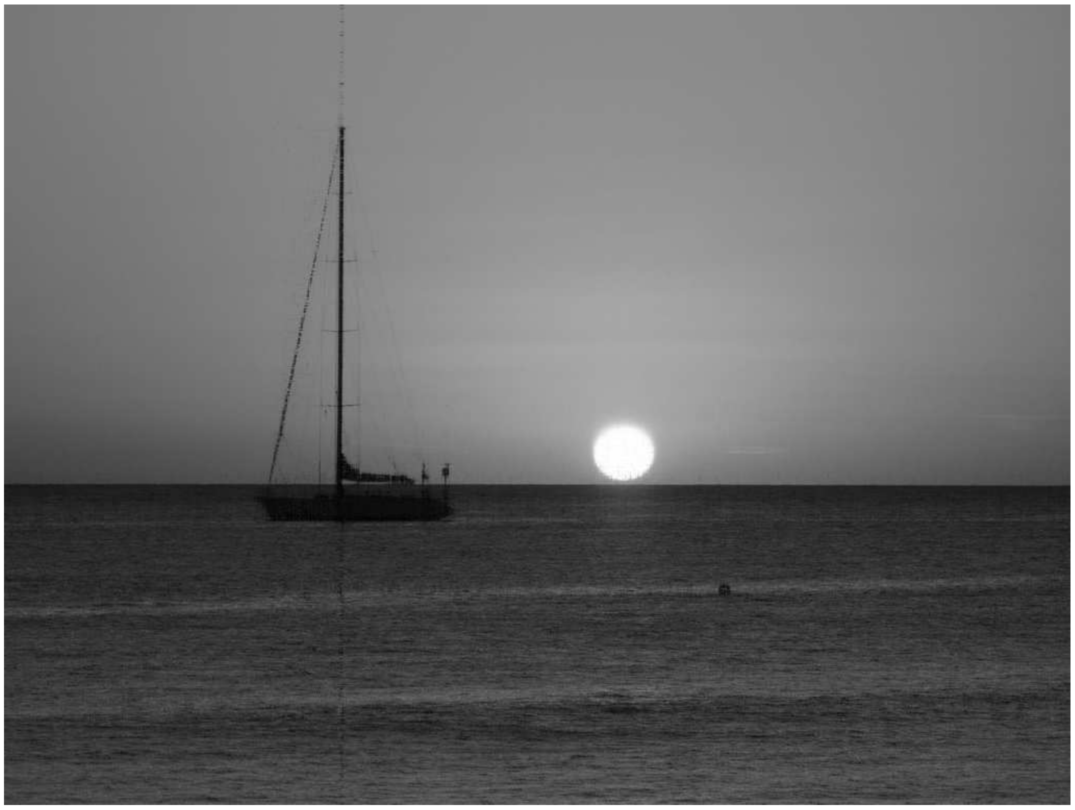


Figure 4.10: Figure recovered by AFPI, rank = 312, error = 4.26×10^3



as well, such that it does not aim at reducing the criterion monotonically. Of course, both clues lead to the same thing in the end. In this case, unlike the previous example, AFPI terminates more lately, but with lower rank and error. Located in a more comparable situation, AFPI still approximates FPILS well even under the assumption violated case.

Chapter 5

Conclusion and Future Work

The thesis studies a family of fixed point iteration algorithms for the purpose of low-rank matrix completion, and provides systematic proofs on the properties of the algorithms. Based on the results, we propose a much more effective fixed point iteration algorithm 4 called AFPI. AFPI not only takes fewer steps to converge, but also forms the objective into the particular structure that is easy to accelerate with truncated SVD.

Beyond the scope of the current work, we are interested to investigate the following topics later in the future.

1. In terms of noisy matrices completion, what is the error upper bound for the fixed point iteration algorithms? For some other methods, there are error upper bound results, e.g., Koltchinskii et al. (2011) and Negahban et al. (2011).
2. The simple rule of updating τ in AFPI algorithm is shown to be very effective in numerical experiments, while it still lacks a rigorous proof. Perhaps there is some relationship with existing methods, or something new requires to be discovered.
3. $O(\frac{1}{k^2})$ is the optimal convergence rate of first-order algorithm, however, the existing method does not well suit the purpose of applying truncated SVD (at least via `PROPACK`). Is there an algorithm fitting to both aspects simultaneously?

4. For recommendation problems, it is often helpful to involve more information, e.g., content of the items. Is there any way to formulate a matrix completion problem to exploit the content information and still keeps its convexity? If yes, then how to?

Appendix A

Convex Analysis

This chapter summarizes the elementary knowledge of convex analysis used in the thesis.

Definition A.1. *An* **optimization problem** *has the form,*

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f_0(x) \\ \text{subject to} \quad & f_i(x) \le b_i, \quad i = 1, \cdots, m, \end{aligned}$$

For a **convex optimization problem***, the functions $f_0, \cdots, f_m : \mathbb{R}^n \to \mathbb{R}$ are all convex, i.e., satisfying*

$$f_i(\alpha x + \beta y) \le \alpha f_i(x) + \beta f_i(y)$$

for all $x, y \in \mathbb{R}^n$ and all $\alpha, \beta \ge 0$ with $\alpha + \beta = 1$.

The significance of convexity in optimization is as essential as linearity in statistics. Generally speaking, a convex optimization problem is relatively easy to solve, as the convexity ensures the local minimum to be the global minimum.

Definition A.2. *g is a* **subgradient** *of f at x if*

$$f(y) \ge f(x) + g^T(y - x) \quad \text{for all } y,$$

and the **subdifferential** *of f at x is the set of all subgradients of f at x*

$$\partial f(x) = \{g : f(y) \geq f(x) + g^T(y - x) \text{ for all } y\}$$

Subgradient (subdifferential) can be viewed as a generalized version of gradient (differential), it is still well defined for some non-smooth function. We can easily verify its scalability and additivity.

Lemma A.3. *Suppose f_1, f_2 are convex functions, and f_2 are continuously differentiable, then it satisfies that*

$$\partial(f_1 + f_2)(x) = \partial f_1(x) + \nabla f_2(x),$$

where $\nabla f_2(x)$ denotes the differential of $f_2(x)$.

Lemma A.3 is quite intuitive: the additivity of subdifferential holds even with differential. Yet the proof is not as simple as it seems, hence we omit it, readers may refer to Nedic and Bertsekas (2003).

Lemma A.4. *A necessary and sufficient condition for $f(x^*)$ to be global minimum is*

$$\mathbf{0} \in \partial f(x^*).$$

Proof. From definition (A.2), the following statements are equivalent

$$\mathbf{0} \in \partial f(x^*) \iff f(y) \geq f(x^*) + \mathbf{0}^T(y - x^*) = f(x^*),$$

which completes the proof. □

Lemma A.4 is intensively used in the proofs. The statement is similar to the optimality condition in differential form. However, the conclusion is much stronger in the sense that, the eligible x^* is a global minimum, compared to the local optimum in differential form. The difference comes from the definition of subdifferential.

Appendix B

SDP Form of Nuclear Norm Minimization

The chapter illustrates how to formulate nuclear norm minimization as an SDP. The result is established on the of dual norm.

Definition B.1. *Given any particular norm of X, i.e., $\|X\|_o$, there exists a corresponding* **dual norm** $\|X\|_d$ *defined as*

$$\|X\|_d := \sup\{\mathrm{Tr}(X^T Y) : \|Y\|_o \leq 1\}$$

Following the definitions, it can be shown that nuclear norm is the dual norm of spectral norm. Moreover, as a byproduct, this gives another illustration of nuclear norm. In this way, we can formulate nuclear norm minimization as an SDP.

Lemma B.2. *The dual norm of spectral norm is the nuclear norm.*

Proof. By definition, the dual norm of spectral norm is the solution of

$$\begin{aligned} \underset{Y}{\text{maximize}} \quad & \mathrm{Tr}(X^T Y) \\ \text{subject to} \quad & \begin{bmatrix} I_m & Y \\ Y^T & I_n \end{bmatrix} \succeq 0, \end{aligned} \tag{B.1}$$

and its dual problem is

$$\begin{aligned} \underset{W_1, W_2}{\text{minimize}} \quad & \frac{1}{2}\left(\mathrm{Tr}(W_1) + \mathrm{Tr}(W_2)\right) \\ \text{subject to} \quad & \begin{bmatrix} W_1 & X \\ X^T & W_2 \end{bmatrix} \succeq 0. \end{aligned} \tag{B.2}$$

Provided the SVD of X, i.e., $X = U\Sigma V^T$, let $W_1 = U\Sigma U^T$, $W_2 = V\Sigma V^T$, it is easy to verify the feasibility of (W_1, W_2) for (B.2), such that

$$\sup_Y \mathrm{Tr}(X^T Y) \leq \frac{1}{2}\left(\mathrm{Tr}(W_1) + \mathrm{Tr}(W_2)\right) = \|X\|_*.$$

From the aspect of primal problem, as $Y = UV^T$ is feasible for (B.1), it follows that

$$\sup_Y \mathrm{Tr}(X^T Y) \geq \mathrm{Tr}(\Sigma) = \|X\|_*.$$

Combining both together yields

$$\mathrm{Tr}(\Sigma) = \|X\|_* \leq \sup_Y \mathrm{Tr}(X^T Y) \leq \|X\|_* = \frac{1}{2}\left(\mathrm{Tr}(W_1) + \mathrm{Tr}(W_2)\right),$$

such that the equality holds throughout, and $\|X\|_*$ is the solution of (B.1), this completes the proof. □

The last step of proof demonstrates that the nuclear norm is the solution of (B.2), which is a typical SDP. For matrix completion problem, the constraint on observed entries can be written in the positive semidefinite matrix form as well. Therefore, the nuclear norm heuristic for matrix completion is still an SDP.

References

Boyd, S. and Vandenberghe, L. (2009). *Convex optimization.* Cambridge university press.

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

Candès, E. J., Ma, Y., Wright, J., et al. (2011). Robust principal component analysis? *Journal of the Association for Computing Machinery*, 58(3).

Candès, E. J. and Plan, Y. (2010). Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936.

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

Candès, E. J. and Tao, T. (2010). The power of convex relaxation: Near-optimal matrix completion. *Information Theory, IEEE Transactions on*, 56(5):2053–2080.

Drineas, P., Kannan, R., and Mahoney, M. W. (2006). Fast monte carlo algorithms for matrices ii: Computing a low-rank approximation to a matrix. *SIAM Journal on Computing*, 36(1):158–183.

Fazel, M. (2002). *Matrix rank minimization with applications.* PhD thesis, Stanford University.

Fazel, M., Hindi, H., and Boyd, S. P. (2001). A rank minimization heuristic with application to minimum order system approximation. In *American Control Conference, 2001*, volume 6, pages 4734–4739. IEEE.

Feuerverger, A., He, Y., and Khatri, S. (2012). Statistical significance of the netflix challenge. *Statistical Science*, 27(2):202–231.

Goldfarb, D. and Ma, S. (2011). Convergence of fixed-point continuation algorithms for matrix rank minimization. *Foundations of Computational Mathematics*, 11(2):183–210.

Grant, M. and Boyd, S. (2014). CVX: Matlab software for disciplined convex programming, version 2.1. http://cvxr.com/cvx.

Gross, D. (2011). Recovering low-rank matrices from few coefficients in any basis. *Information Theory, IEEE Transactions on*, 57(3):1548–1566.

Hale, E. T., Yin, W., and Zhang, Y. (2007). A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing. *CAAM TR07-07, Rice University*.

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

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The elements of statistical learning, 2nd edition*. Springer.

Hu, Y., Zhang, D., Ye, J., Li, X., and He, X. (2013). Fast and accurate matrix completion via truncated nuclear norm regularization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(9):2117–2130.

Jain, P., Meka, R., and Dhillon, I. S. (2010). Guaranteed rank minimization via singular value projection. In *Advances in Neural Information Processing Systems*, pages 937–945.

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

Keshavan, R., Montanari, A., and Oh, S. (2009). Matrix completion from noisy entries. In *Advances in Neural Information Processing Systems*, pages 952–960.

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

Koltchinskii, V., Lounici, K., Tsybakov, A. B., et al. (2011). Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion. *The Annals of Statistics*, 39(5):2302–2329.

Lai, M.-J. and Yin, W. (2013). Augmented ℓ_1 and nuclear-norm models with a globally linearly convergent algorithm. *SIAM Journal on Imaging Sciences*, 6(2):1059–1091.

Larsen, R. M. (1998). Lanczos bidiagonalization with partial reorthogonalization. *DAIMI Report Series*, 27(537).

Lin, Z., Chen, M., and Ma, Y. (2010). The augmented lagrange multiplier method for exact recovery of corrupted low-rank matrices. *arXiv preprint arXiv:1009.5055*.

Ma, S., Goldfarb, D., and Chen, L. (2011). Fixed point and bregman iterative methods for matrix rank minimization. *Mathematical Programming*, 128(1-2):321–353.

Mazumder, R., Hastie, T., and Tibshirani, R. (2010). Spectral regularization algorithms for learning large incomplete matrices. *Journal of Machine Learning Research*, 11:2287–2322.

Nedic, A. and Bertsekas, D. (2003). *Convex analysis and optimization*. Athena Scientific.

Negahban, S., Wainwright, M. J., et al. (2011). Estimation of (near) low-rank matrices with noise and high-dimensional scaling. *The Annals of Statistics*, 39(2):1069–1097.

Recht, B. (2011). A simpler approach to matrix completion. *The Journal of Machine Learning Research*, 12:3413–3430.

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

Rohde, A., Tsybakov, A. B., et al. (2011). Estimation of high-dimensional low-rank matrices. *The Annals of Statistics*, 39(2):887–930.

Srebro, N., Rennie, J., and Jaakkola, T. S. (2004). Maximum-margin matrix factorization. In *Advances in neural information processing systems*, pages 1329–1336.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, pages 267–288.

Vandenberghe, L. and Boyd, S. (1996). Semidefinite programming. *SIAM review*, 38(1):49–95.

Watson, G. A. (1992). Characterization of the subdifferential of some matrix norms. *Linear Algebra and its Applications*, 170:33–45.

Wen, Z., Yin, W., Zhang, H., and Goldfarb, D. (2012). On the convergence of an active-set method for ℓ_1 minimization. *Optimization Methods and Software*, 27(6):1127–1146.

Wright, J., Ganesh, A., Min, K., and Ma, Y. (2013). Compressive principal component pursuit. *Information and Inference*, 2(1):32–68.

Wright, J., Ganesh, A., Rao, S., Peng, Y., and Ma, Y. (2009). Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization. In *Advances in neural information processing systems*, pages 2080–2088.

Zhou, Z., Li, X., Wright, J., Candès, E., and Ma, Y. (2010). Stable principal component pursuit. In *Information Theory Proceedings, 2010 IEEE International Symposium*, pages 1518–1522. IEEE.