Distribution Agreement

In presenting this thesis as a partial fulfillment of the requirements for a degree from Emory University, I hereby grant to Emory University and its agents the non-exclusive license to archive, make accessible, and display my thesis in whole or in part in all forms of media, now or hereafter now, including display on the World Wide Web. I understand that I may select some access restrictions as part of the online submission of this thesis. I retain all ownership rights to the copyright of the thesis. I also retain the right to use in future works (such as articles or books) all or part of this thesis.

Signature:

Jiasheng Sheng April 1, 2022

A Comparison Study on Low Rank Matrix Completion Algorithms in Image and Painting Restoration

By

Jiasheng Sheng

James Nagy Ph.D.
Advisor

Department of Mathematics

James Nagy Ph.D.
Advisor

Joyce Ho, Ph.D.
Committee Member

Li Xiong, Ph.D.
Committee Member

2022

A Comparison Study on Low Rank Matrix Completion Algorithms in Image and Painting Restoration

By

Jiasheng Sheng

James Nagy Ph.D.
Advisor

An abstract of
a thesis submitted to the Faculty of Emory College of Arts and Sciences of
Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Mathematics

2022

Abstract

A Comparison Study on Low Rank Matrix Completion Algorithms in Image and Painting Restoration
By Jiasheng Sheng

Low rank matrix completion solves prevalent real life problems including image restoration, movie recommendation and photo depth enhancement. However, there is not a systematic comparison and evaluation for different low rank matrix completion algorithms. This thesis aims to focus on painting restoration and analyze the running time and relative Frobenius norm for three different algorithm: Singular Value Thresholding (SVT), Iteratively Reweighted Least Squares (IRLS) and Low-rank Matrix Fit (LMaFit). Using custom testing images generated from a publicly available dataset of contemporary abstract paintings, it is observed that the LMaFit algorithm has the fastest running time but the worst accuracy. SVT has similar accuracy as IRLS but has faster running time. However, IRLS has more options for parameter tuning than SVT, especially the rank of desired matrix. So each algorithm has its own benefits and drawbacks in terms of image restoration or denoising.

A Comparison Study on Low Rank Matrix Completion Algorithms in Image and Painting Restoration

By

Jiasheng Sheng

James Nagy Ph.D.
Advisor

A thesis submitted to the Faculty of Emory College of Arts and Sciences
of Emory University in partial fulfillment
of the requirements of the degree of
Bachelor of Science with Honors

Department of Mathematics

2022

Acknowledgments

Emory University offered me great opportunities in my three years of study as a transfer student. In Emory, I was fortunate to have Dr. James Nagy as my thesis advisor. The first semester of my study in Emory University, I took a Linear Algebra class taught by Dr. Nagy, and I had great fun pondering on class materials. Dr. Nagy was able to truly deliver lectures with his own understanding of materials, and he wanted his students to understand and have fun with linear algebra with his passion. One year later, in Dr. Nagy's Numerical Analysis class, I realized my interested in applied numerical linear algebra and wanted to do an honor's thesis. After several meetings, my journey started. I am truly thankful for Dr. Nagy for giving me ideas to write this thesis and for showing me great passion in applied math research.

In the process of writing this thesis, many other classes I took gave me great ideas as well. In the Machine Learning class by Dr. Li Xiong, I learned a lot about machine learning basics such as gradient descent and many classification algorithms. I want to thank Dr. Xiong for not only leading me exploring the world of machine learning but also for being a part of the thesis committee. I also want to thank Dr. Joyce Ho for being a part of the committee and as a great research advisor. I have been fortunate to work with Dr. Ho on an NLP project. With Dr. Ho's guidance, I not only learned numerous knowledge in NLP but I also further consolidated my interest in machine learning. Most importantly, I cannot express more thankfulness towards my parents for supporting my whole undergraduate study in the United States. Without them, I can never discover my interest in applied math or be the person who I am right now.

Contents

List of Figures

List of Algorithms

Chapter 1

Introduction

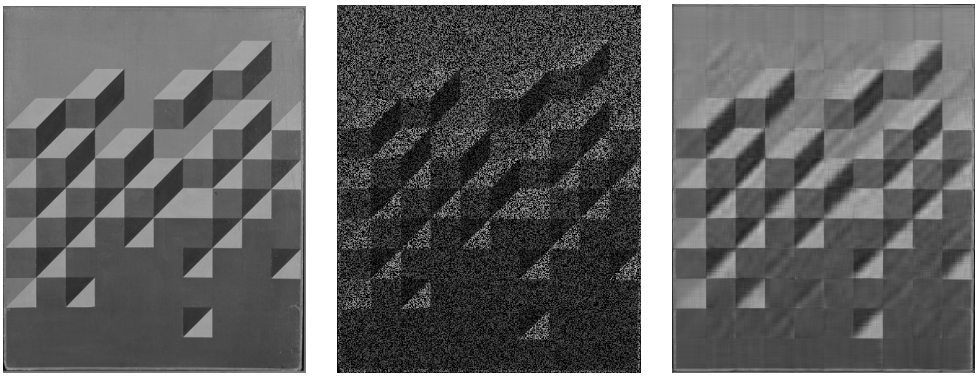
Matrix completion, by its name, means to fill out the empty entries in a given matrix. Low-rank matrix completion is to fill out the empty entries in a matrix while minimizing the rank of the completed matrix. This chapter will introduce the reason for the low-rank constraint and some real-life applications on low-rank matrix completion. Later in this chapter, some previous studies will be introduced. At last, the dataset used in this thesis and some data preprocessing methods will be discussed.

1.1 Introduction

Low rank matrix completion problems are more prevalent in our life than we can imagine. One of the most prominent examples is the 2006 Netflix Prize of 1 million dollars for the first person or team who could increase by at least 10% the performance of Netflix's recommendation system [22]. The problem setup was a large matrix with each column representing a user and each row representing that user's rating for all Netflix movies. This is a huge matrix with millions of users and tens of thousands of movies; the matrix is also composed with largely empty spaces because it is impossible for a user to watch every single movie on Netflix. The goal is to predict a user's rating for an unseen movie to better give recommendations to users. The only logical way is

to try to group users with similar movie tastes and then predict a new rating based on existing users' ratings. The low rank constraint becomes apparent for the need of linearly dependent columns if we want to match as many similar columns as possible. One of the solutions for the Netflix prize is using matrix factorization, which projects a large problem into a smaller space [28].

The low rank matrix completion problem is also applied in image processing such as image and video denoising [4, 13] because we want to complete a matrix without noise. Figure 1.1 shows an example of a recovered image after applying the Singular Value Thresholding algorithm (which will be described in Chapter 2) to a given noisy image.



Original picture 50% data left Recovered picture

Figure 1.1: A sample demo of matrix completion with Singular Value Thresholding Algorithm.

Low-rank matrix completion not only works in dealing with noise, it also works with image inpainting [32], which means trying to recover a picture where there are more contiguous missing pixels instead of randomly distributed errors due to noise. Some other applications include enhancing the depth of a camera shot image [19]. Another interesting application of low-rank matrix completion is to predict the emotion in abstract painting by trying to recover a special block matrix that contains a feature matrix of an image and its corresponding recognized emotion matrix [1].

1.2 Problem Setup

With the understanding of why it may be important to impose a low rank constraint on the matrix completion problem, the problem can be set up with the equation below:

$$\begin{aligned} &\text{minimize } rank(X) \\ &\text{subject to } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned} \tag{1.1}$$

X is the recovered matrix we want to compute and M is the noisy matrix with initial data. The operator $\mathcal{P}_\Omega(X)$ means to select all entries in matrix X in the region of Ω. In this case, $\mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M)$ simply means the given entries in data matrix M should be the same in the recovered matrix X with the region Ω being the matrix indices of given data.

One brute force way to solve this problem is to try to find a valid solution by iterating over all possible ranks [23]. Specifically, start with assuming the original matrix is rank 1 (i.e., $rank(M) = 1$), and then try to solve the system with all linearly dependent columns $M_i = c \cdot M_j$. If no solution is found, then assume $rank(M) = 2$ and then try to solve the system with $M_i = c \cdot M_j + d \cdot M_k$. However, this brute force approach requires exponential running time, and is not feasible for a large problem. The time complexity is $\mathcal{O}(n2^n)$ for an n by n matrix because it takes n iterations for maximum $rank = n$ and there are $\mathcal{O}(2^n)$ computations in each iteration. Problem (1.1) by itself is a difficult non-convex problem to solve, so some transformation is required to make the problem tractable. The most researched heuristic for the low rank minimization problem is to minimize the nuclear norm of the matrix [23]. The nuclear norm $\|\cdot\|_*$ is the sum of all singular values of a matrix, and the nuclear norm is also a convex function that can be optimized efficiently [26]. The optimization

problem takes the form:

$$\begin{aligned} &\text{minimize } \|X\|_* \\ &\text{subject to } \mathscr{P}_\Omega(X) = \mathscr{P}_\Omega(M) \end{aligned} \tag{1.2}$$

This problem setup gives rise to the following three algorithms that are going to be discussed in this thesis: Singular Value Thresholding (SVT) [2], Iterativly Reweighted Least Square (IRLS) [5] and Low-rank Matrix Fit (LMaFit) [31].

There is another heuristic that transforms the rank minimization problem, which is to minimize the residual Frobenius-norm of the completed matrix. The Frobenius norm $\|\cdot\|_F$ is the square root of the sum of entries squared, $\|A\|_F = \sqrt{\sum_i \sum_j A_{ij}^2}$. So this problem setup is:

$$\text{minimize}_X \|\mathscr{P}_\Omega(M) - \mathscr{P}_\Omega(X)\|_F^2. \tag{1.3}$$

This setup can be solved by first factoring X into two smaller dimension matrices and then applying a robust gradient descent algorithm for the optimization problem [20]. There is also a study that uses manifold learning with a deep learning approach to solve this Frobenius-norm setup as well [21].

This paper is going to focus on the minimize nuclear norm setup (1.2). The performance for each algorithm will be measured in terms of error and running time, and the error evaluation for the result will be the relative Frobenius-norm residual, which can be defined as error $= \frac{\|I-X\|_F}{\|I\|_F}$ where I is the original matrix without noise and X is the recovered matrix.

The focus of this paper is image denoising, more specifically on images of paintings. Some old paintings that were not maintained properly will have small cracks and imperfections, which can be treated as noise. There are already many studies on painting restoration in the field of signal processing [24, 8]. Specifically, the

texture and paint stroke can be restored without being blurred using a patch based anisotropic-diffusion technique [25]. However, previous studies on painting restoration lacks the use of matrix completion techniques. Thus, investigating on applying matrix completion algorithms for painting restoration is an interesting application to look into.

1.3 Dataset

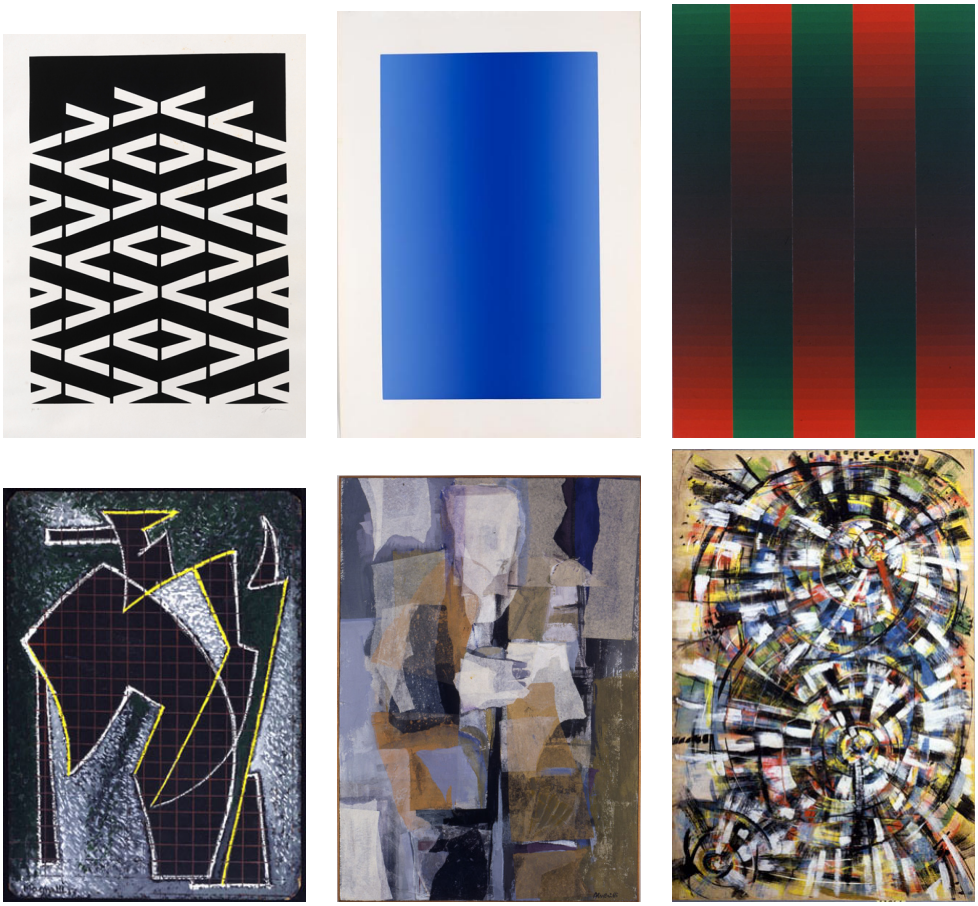


Figure 1.2: Sample simple (top) and hard (bottom) pictures from *MART dataset*.

This study uses a publicly available dataset containing images of paintings from the Museum of Modern and Contemporary Art of Trento and Rovereto (MART), hence the name *MART dataset* [33]. The *MART dataset* contains a selection of 500 con-

temporary paintings from the beginning of 20th century. All paintings in the dataset is created by a mixture of over 80 professor artists. All of the paintings are abstract paintings, meaning there is a variety of shapes and texture on each painting that can either be structured or unstructured.

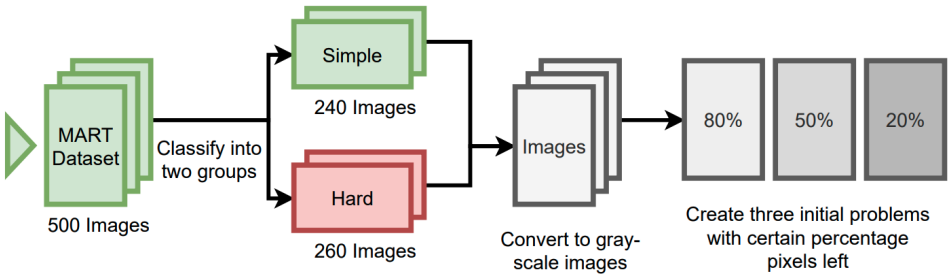
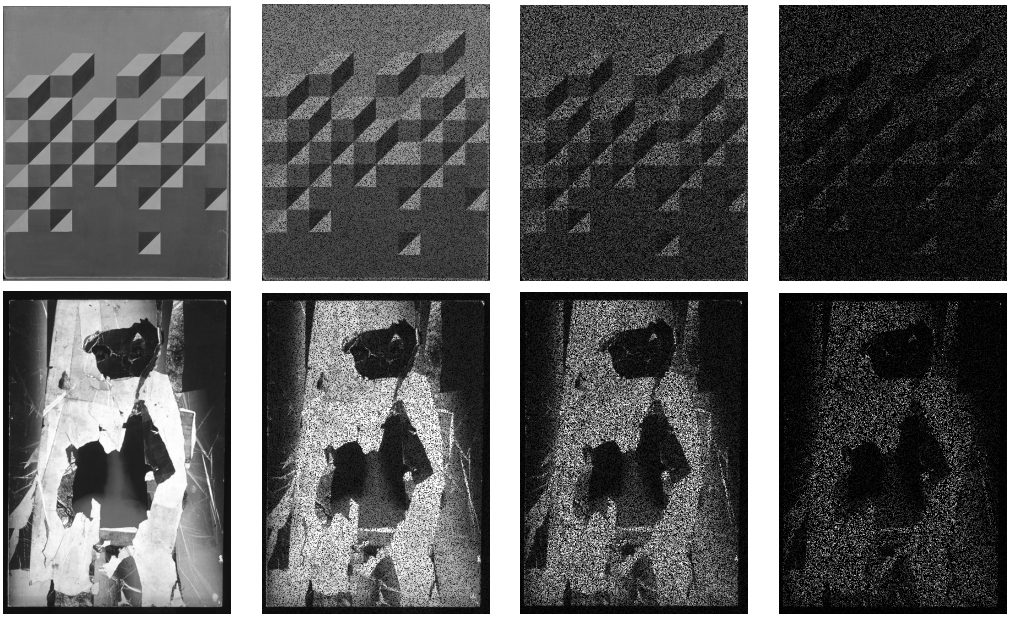


Figure 1.3: Flow chart of data processing.

All images from the *MART dataset* are colored images that have three color channels, red green and blue (RGB), which need to be converted to greyscale data matrix with one channel of floating point numbers. Also there is no need to rescale each image because most images are less than 1000*1000 pixels, which is small enough for doing numerical experiments efficiently. To test each algorithm's performance on different problems, each test case generates three "observed" matrices with only 80%, 50% and 20% of the total pixels left from the original image (for each image, a certain percentage of randomly selected elements are changed to 0). So in theory, a matrix with 80% of the total data should be easier to recover than a matrix with only 20% of the total data. Due to the nature of abstract painting, the images in the *MART dataset* were manually divided into two classes: simple and hard. We loosely define simple to mean there is a clear structure of the painting or there are a lot of repeated patterns; hard images mean there is no clear structure and the composition of painting is mostly random. Each algorithm will be tested separately for simple and hard images. In total, there are 240 images in the simple group and 260 images in the hard group.

To show the result in a visual way, the two images below will be displayed as the sample picture in this thesis.



Original 80% remaining 50% remaining 20% remaining

Figure 1.4: Sample simple (top) and hard (bottom) demonstration images.

Chapter 2

Singular Value Thresholding

The general formulation of a convex reduced problem for matrix rank minimization is the nuclear norm minimization, which is introduced in chapter 1 and can be written as the below format

$$\begin{aligned} &\text{minimize } \|X\|_* \\ &\text{subject to } \mathscr{P}_\Omega(X) = \mathscr{P}_\Omega(M) \end{aligned}$$

The singular value thresholding (SVT) algorithm is based on singular value decomposition (SVD). The SVD of a matrix X is $X = U\Sigma V^\top$. For an $n_1 \times n_2$ matrix X, U is an $n_1 \times n_1$ orthogonal matrix, V is an $n_2 \times n_2$ orthogonal matrix and Σ is a diagonal matrix with singular values on the diagonal. The SVT algorithm uses a soft thresholding on the singular values on a matrix in each iteration. Let $X = U\Sigma V^\top$, and the threshold operator $\mathscr{D}$ on matrix X is defined in the original paper [2]:

$$\mathscr{D}_\tau(X) = U\mathscr{D}_\tau(\Sigma)V^\top, \;\; \mathscr{D}\tau(\Sigma) = diag(\{\sigma_i - \tau\}_+), \tag{2.1}$$

where the operator $\{c\}_+$ is defined as $\{c\}_+ = max(c, 0)$. Thus, if the value $\sigma_i - \tau$ is smaller than τ, the value is set to zero. So in essence, the operator $\mathscr{D}$ shrinks the singular values of a matrix.

The iteration step uses two matrices X and Y, the threshold τ and a step size δ_k

that varies for each iteration,

$$\begin{aligned} X^k &= \mathcal{D}_\tau(Y^{k-1}) \\ Y^k &= Y^{k-1} + \delta_k \mathcal{P}_\Omega(M - X^k) \end{aligned} \tag{2.2}$$

and this iterative step converges to a new problem setup, and the actual proof is in section 4 of the original paper [2]:

$$\begin{aligned} &\text{minimize } \tau\|X\|_* + \tfrac{1}{2}\|X\|_F^2 \\ &\text{subject to } \mathcal{P}_\Omega(X) = \mathcal{P}_\Omega(M) \end{aligned} \tag{2.3}$$

An alternative form of problem (2.3) can be represented via a Lagrangian approach, which offers another intuition to the original problem [15]. It can also be shown that (2.3) is able to converge to the original problem (1.2) when $\tau \to \infty$. An intuitive way to view this is that problem (2.3) divides the problem with a weighted sum for nuclear norm and Frobenuis-norm. If the weight for the nuclear norm is significantly larger than the weight of the Frobenuis-norm, the problem is essentially the nuclear norm setup.

2.1 Choosing the Threshold

The choice of τ is another optimization problem to solve [18], and there are many different perspectives on choosing the value of τ: a fixed choice of $\tau = \frac{4}{\sqrt{3}}$ was said to have an optimal performance for a non-square matrix [6], and another choice of τ is the minimum difference in the singular values [27]. In the original paper [2], τ was set as $5 \cdot \sqrt{n_1 \cdot n_2}$, with $[n_1, n_2]$ = size of(M), and if there is a large difference between n_1 and n_2, τ should be larger than $5 \cdot \sqrt{n_1 \cdot n_2}$. Since a large τ in 2.3 means the problem has less weight on the Frobenius norm and more weight on the nuclear norm, the problem will be closer to 1.2 but with more error in computation since the

singular values will shrink faster. Thus in general, the choice of threshold is between $2 \cdot \sqrt{n_1 \cdot n_2}$ to $5 \cdot \sqrt{n_1 \cdot n_2}$.

Although parameter tuning was not introduced in the original paper, in the actual implementation, τ is a parameter that can be tuned to achieve maximum performance for the SVT algorithm. Maybe there are ways to even vary the value of τ in each iteration to fit problems better.

2.2 Choosing the Step Size and Stopping Criteria

The choice of step size δ can vary in each iteration, or it can be a fixed value for a simpler computation. The iteration converges at step k as long as the step size δ satisfies the following constraint [2]:

$$0 < \delta < 2 \cdot \frac{\|X^\star - X^k\|_F^2}{\|\mathcal{P}_\Omega(X^\star - X^k)\|_F^2} \tag{2.4}$$

Matrix $X^\star$ is the solution to the problem (2.3); that is the minimization of $\tau\|X\|_* + \frac{1}{2}\|X\|_F^2$. Too small of a step size will compromise the convergence speed, so a moderate size $\delta = 1.2 \cdot p^{-1}$ is recommended, where $p = \frac{\text{\# of known entries}}{\text{total \# of entries}}$ is the percentage of known entries. You can always decrease the step size and run more iterations if the result does not converge.

Stopping criteria is chosen as a tolerance for the relative Frobenius-norm residual

$$\frac{\|\mathcal{P}_\Omega(X^k - M)\|_F}{\|\mathcal{P}_\Omega(M)\|_F} < \epsilon \tag{2.5}$$

where M is the input matrix with known data, X^k is the approximation of X computed at step k, and ϵ is the stopping tolerance.

2.3 Algorithm

Putting the iterative steps (2.2) in an actual algorithm format, we obtain

input : data matrix M, initial step size δ, threshold τ, tolerance ϵ

output: completed matrix X^k

initialization: stopping flag $flag = False$, counter $k = 0$ and $Y^0 = 0$;

while *!flag* **do**

$k = k + 1$;

$[U^{k-1}, \Sigma^{k-1}, V^{\top k-1}] = \texttt{svd}(Y^{k-1})$;

$X^k = U^{k-1}\mathscr{D}_\tau(\Sigma^{k-1})V^{\top k-1}$;

$Y^k = Y_{k-1} + \delta(\mathscr{P}_\Omega(M - X^k))$;

$flag = \frac{\|\mathscr{P}_\Omega(X^k - M)\|_F}{\|\mathscr{P}_\Omega(M)\|_F} < \epsilon$;

end

Algorithm 1: SVT Algorithm

The limitation of this algorithm is that it computes an SVD of matrix Y, which is slow when Y is large. Therefore, many other SVT based algorithms treat computing the value of a matrix by the threshold operator (2.1) as a new problem [11]. One easier solution is to try to accelerate the process of computing only a partial singular value decomposition. For example, using the Lanczos SVD from PROPACK [16] to compute the approximated SVD with smaller rank is faster than computing the whole SVD, especially in a low rank setup. Another solution is to use polar decomposition and matrix Newton iteration [3], but this approach is too complicated and is not actually helpful with the accuracy of solving the matrix completion problem.

On the other hand, the benefit for the SVT algorithm is that it does not require a guessed rank as input to the algorithm for the final completed matrix. For matrix

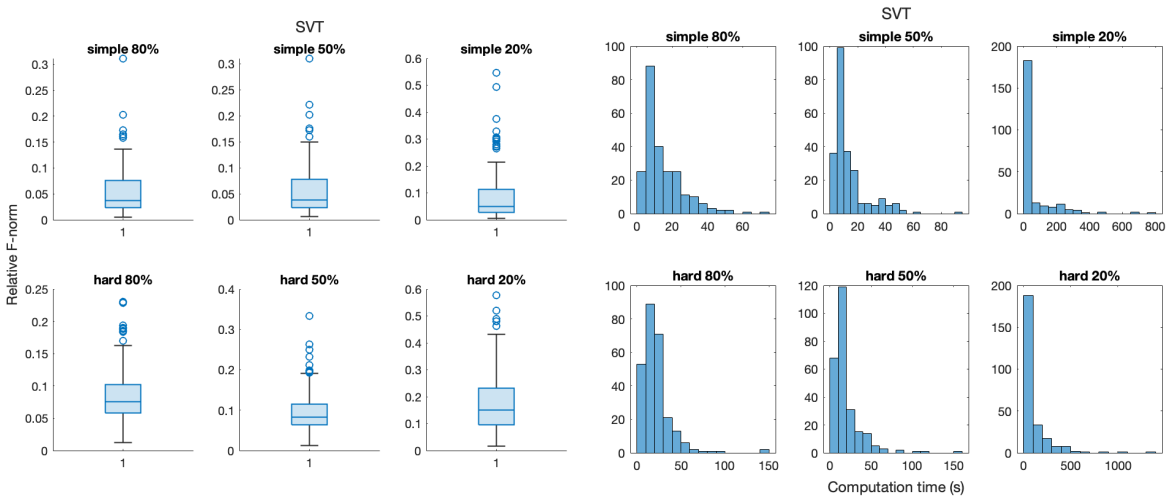


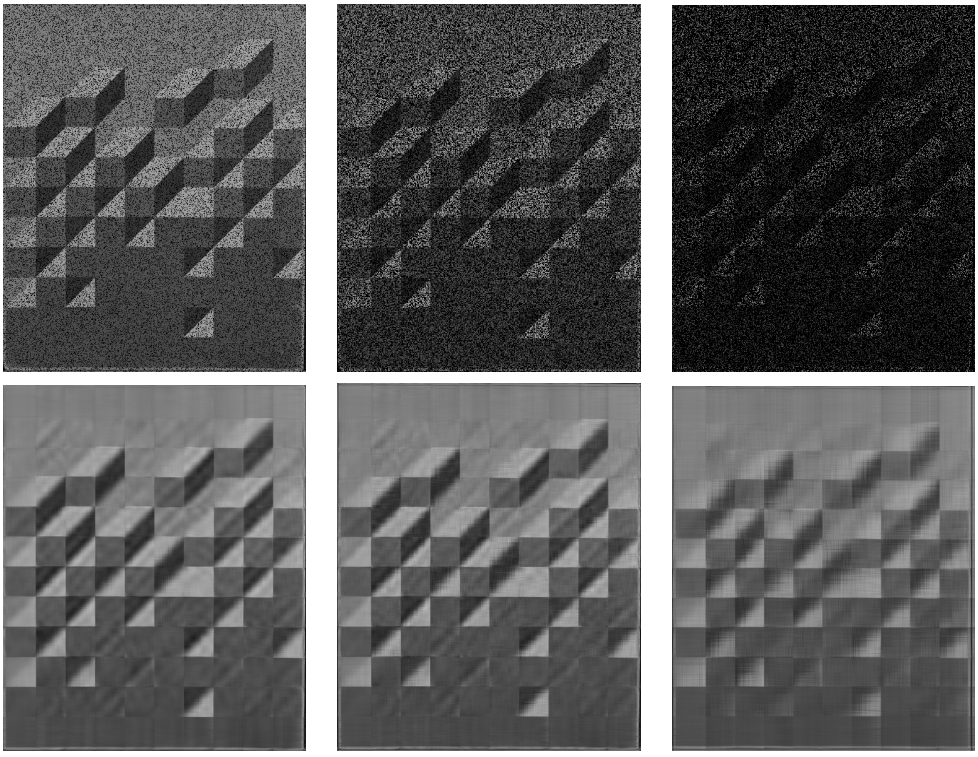
Figure 2.1: Relative residual Frobenius-norm (right) and run time (right) for test images for the SVT algorithm. The upper row shows the result for simple test images and bottom row shows the result for hard test images. Each section has three columns. The left most column shows the result for images with 80% remaining pixels, the middle shows result with 50% remaining and the right shows result with 20% remaining.

factorization based methods, a rank is often required as an input to convert a large matrix into two smaller matrices. In most cases, you do not know exactly what the rank will be, so a guessed value as input to the algorithm might be inaccurate. Of course, in some cases, there is a desired rank for the output matrix, and the SVT algorithm does not have control in this situation.

2.4 Numerical Experiments

Using images from the *MART dataset*, each image was preprocessed with the steps in 1.3. The machine used to conduct this numerical experiment is a 2017 MacBook Pro with 16G memory. From Figure 2.1, it can be observed that the running time for the SVT algorithm is fast. Images in the *MART dataset* are relatively small, so the time for computing the SVD does not have a strong factor in the total running time. There are some extreme values that take more than 500 seconds for simple and hard

cases with only 20% of pixels left. The pattern for running time is that it generally increases as the number of known entries decreases or as the problem becomes harder and less structured. The relative residual Frobenius-norm is small for simple and hard images with 80% and 50% data. The mean is less than 0.1, although there are some values that are greater than 0.5. In general, even for hard problems with only 20% pixels left, the SVT algorithm works well.

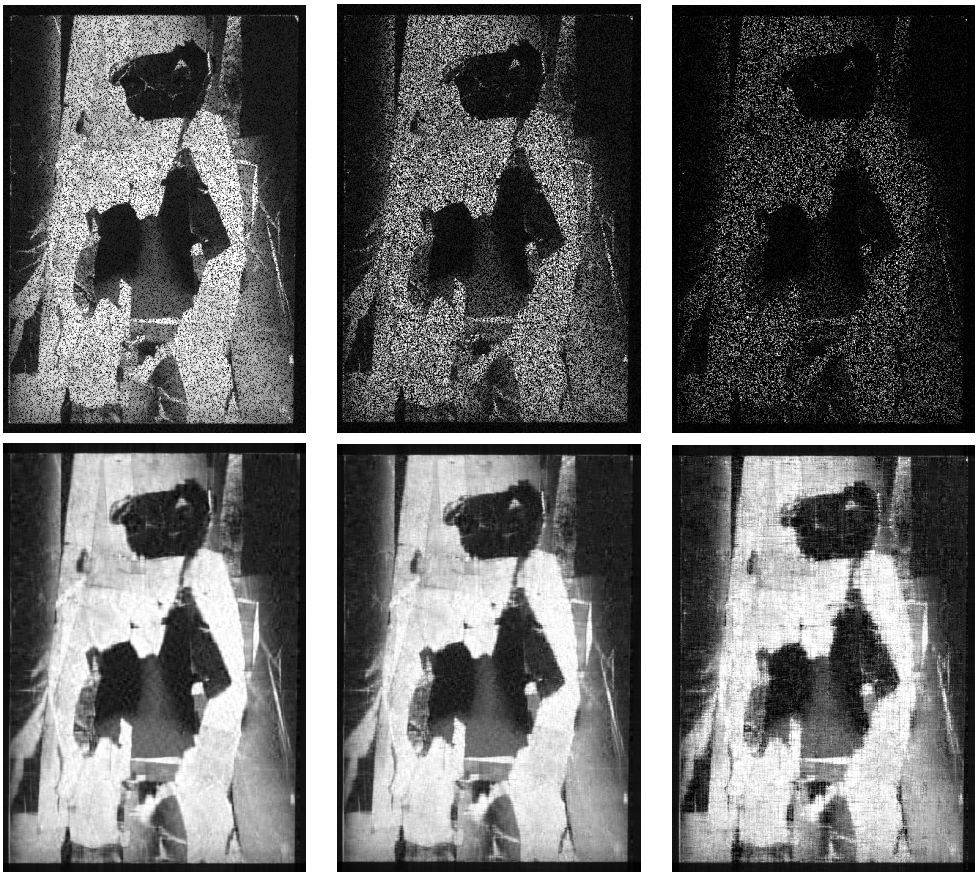


80% remaining 50% remaining 20% remaining

Figure 2.2: The original and recovered of a simple problem with 80%, 50% and 20% remaining with the SVT algorithm.

With the actual recovered picture for a specific simple problem shown in Figure 2.2, we can observe that the recovered picture with only 20% of the data left loses certain structure detail, as the diagonal edges become blurred. In the recovered picture for a specific hard problem shown in Figure 2.3, the recovered picture with 80% and 50% of the data have similar results. This experiment does not include parameter tuning for the threshold τ or the step-size δ, which can be another area to

look into in the future.



80% remaining 50% remaining 20% remaining

Figure 2.3: The original and recovered of a hard problem with 80%, 50% and 20% remaining with the SVT algorithm.

Chapter 3

Iteratively Reweighted Least Squares

The Iteratively Reweighted Least Squares (IRLS) algorithm is based on the problem of minimizing the nuclear norm, similiar to the setup in the SVT algorithm (1.2),

$$\begin{aligned} &\text{minimize } \|X\|_* \\ &\text{subject to } \mathscr{P}_\Omega(X) = \mathscr{P}_\Omega(M) \end{aligned} \tag{3.1}$$

The whole intuition for IRLS is to transform the nuclear norm in (3.1) into a Forbenius-norm, which can in-turn change into a 2-norm least squares problem. So the first step is to envision a Frobenius-norm from the nuclear norm.

In the original paper [5], the Schatten q-norm is used, which is defined by:

$$\|X\|_{S_q} = \|\sigma(X)\|_{l_q^n} = (\sum_{i=1}^{n} |\sigma_i(X)|^q)^{1/q} \tag{3.2}$$

It is obvious that when $q = 1$, the Schatten q-norm is the same as the nuclear norm,

which is just the sum of all singular values.

$$\|X\|_{S_1} = \sum_{i=1}^{n} \sigma_i(X) = \|X\|_* \tag{3.3}$$

The next step is to make a connection between trace, which is in the Frobenius-norm since $\|X\|_F = Tr(XX^\top)^{1/2}$ and singular values, which is in the Schatten q-norm. We want to show

$$\|X\|_{S_q} = (Tr(|X|^q))^{1/q} \text{ where } |X| = (XX^\top)^{1/2} \tag{3.4}$$

Proof. Let $X = U\Sigma V^\top$. $|X| = (U\Sigma V^\top V\Sigma U^\top)^{1/2} = (UD^2U^\top)^{1/2}$ where $D = diag(\sigma_i)$.

$$|X| = (UD^2U^\top)^{1/2} = (UDDU^\top)^{1/2} = (UDU^\top UDU^\top)^{1/2} = UDU^\top$$

$$|X|^q = UD^qU^\top$$

$$Tr(|X|^q) = Tr(D^q) = \sum_i \sigma_i^q$$

$$\text{From (3.2), } \|X\|_{S_q} = (\sum_i^n |\sigma_i(X)|^q)^{1/q} = (Tr(|X|^q))^{1/q}$$

□

From equation (3.4), and (3.3)

$$\|X\|_* = \|X\|_{S_1} = Tr((XX^\top)^{1/2}) = Tr((XX^\top)^{-1/2}(XX^\top)) = \|W^{1/2}X\|_F^2, \tag{3.5}$$

where $W = (XX^\top)^{-1/2}$.

Then the problem of minimizing a nuclear norm becomes minimizing the Frobenius-norm. The following iterative step can be defined as we update X and W in each

iteration:

$$\begin{aligned} X^{k+1} &= \underset{X}{\text{minimize}} \, \|(W^k)^{1/2} X\|_F^2 \\ &\text{subject to } \mathscr{P}_\Omega(X) = \mathscr{P}_\Omega(M) \\ W^{k+1} &= (X^{k+1}(X^{k+1})^\top)^{-1/2} \end{aligned} \tag{3.6}$$

3.1 Least Squares

To solve the minimization problem in (3.6), many methods can be used to transform the Frobenius-norm into a 2-norm minimization least squares problem. One method uses the Kronecker product. The Kronecker product $\otimes$ for two matrices X with size $n_1 \times n_2$ and Y with size $m_1 \times m_2$ is defined as

$$X \otimes Y = \begin{bmatrix} X_{1,1}Y & X_{1,2}Y & \cdots & X_{1,n_2}Y \\ X_{2,1}Y & X_{2,2}Y & & \\ \vdots & & \ddots & \\ X_{n_1,1}Y & & & X_{n_1,n_2}Y \end{bmatrix}$$

The result is a large matrix with size $n_1 m_1 \times n_2 m_2$.

$$\|WX\|_F^2 = \left\| \begin{bmatrix} W & 0 & \cdots & 0 \\ 0 & W & & \\ \vdots & & \ddots & \\ 0 & & & W \end{bmatrix} \cdot \boldsymbol{X} \right\|_2^2 = \|(I \otimes W) \cdot \boldsymbol{X}\|_2^2$$

where $\boldsymbol{X}$ is vector with all columns of X stretched into a vector. $\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X_1} \\ \boldsymbol{X_2} \\ \vdots \\ \boldsymbol{X_n} \end{bmatrix}$. $\boldsymbol{X_i}$ denotes the i^{th} column of matrix X. This is only a rudimentary approach in solving

this problem since explicitly forming the Kronecker product of large matrices can take up to much memory and is not efficient in computation. In some research, conjugate gradient is widely adopted in solving this problem [12] along with the use of quotient singular value decomposition and generalized inverses [29]. In iterative methods, like conjugate gradient, the structure of Kronecker products can be efficiently exploited.

3.2 Choosing Threshold

In the iterative step (3.6), matrix W is updated in each iteration by matrix X. If X is ill-conditioned, there might be a large error in the computation for each iteration. Thus, a constraint on singular values must be put into matrix X to ensure that it is not ill-conditioned. The threshold τ, which is similar to (2.1), but it is updated in each iteration, instead of being a fixed value in the SVT algorithm. The approach proposed in the IRLS paper [5] is

$$\tau^k = \min \{\tau^{k-1}, \gamma\sigma_L(X^k)\} \tag{3.7}$$

in the k^{th} iteration where $\gamma = \frac{1}{n_1}$, n_1 is the number of rows of matrix X, and $\sigma_L(X^k)$ is the L^{th} singular value of the current X where L is a constant that is larger than the desired rank for the completed matrix: $L \geq r$, r is the desired rank. The proof of this setup is in section 2.3 of the IRLS paper [5]. The parameter L can be tuned to maximize the performance of the algorithm but with the problem of unknown initial rank.

After computing the threshold for each iteration, the update process for W is evident. Because $W = (XX^\top)^{-1/2}$, and let $U\Sigma V^\top$ be the singular value decomposition

for matrix X, then assuming X is an $n \times n$ square matrix

$$\begin{aligned} XX^\top &= U\Sigma^2 U^\top \\ W &= (XX^\top)^{-1/2} \\ \mathscr{D}_\tau(W) &= U(\Sigma_\tau)^{-1}U^\top, \end{aligned} \tag{3.8}$$

where Σ_τ is the threshold operator for making all diagonal entries of Σ zeros if the value is smaller than τ.

3.3 Algorithm

To sum up the previously discussed iterative steps, we obtain the following algorithm:

input : data matrix M, initial $W = I$, threshold τ, tolerance ϵ

output: completed matrix X^k

initialization: stopping flag $flag = False$, counter $k = 0$, and $X^0 = M$;

while *!flag* **do**

- $k = k + 1$;
- $X^k = \text{argmin}_{\mathscr{P}_\Omega(X)=\mathscr{P}_\Omega(M)} \|(W^{1/2})^{k-1}X^{k-1}\|_F^2$;
- update τ with method of 3.7 ;
- $[U, \Sigma, V^\top] = \texttt{svd}(X)$;
- $W^k = U(\Sigma_\tau)^{-1}U^\top$;
- $flag = \frac{\|\mathscr{P}_\Omega(X^k - M)\|_F}{\|\mathscr{P}_\Omega(M)\|_F} < \epsilon$;

end

Algorithm 2: IRLS Algorithm

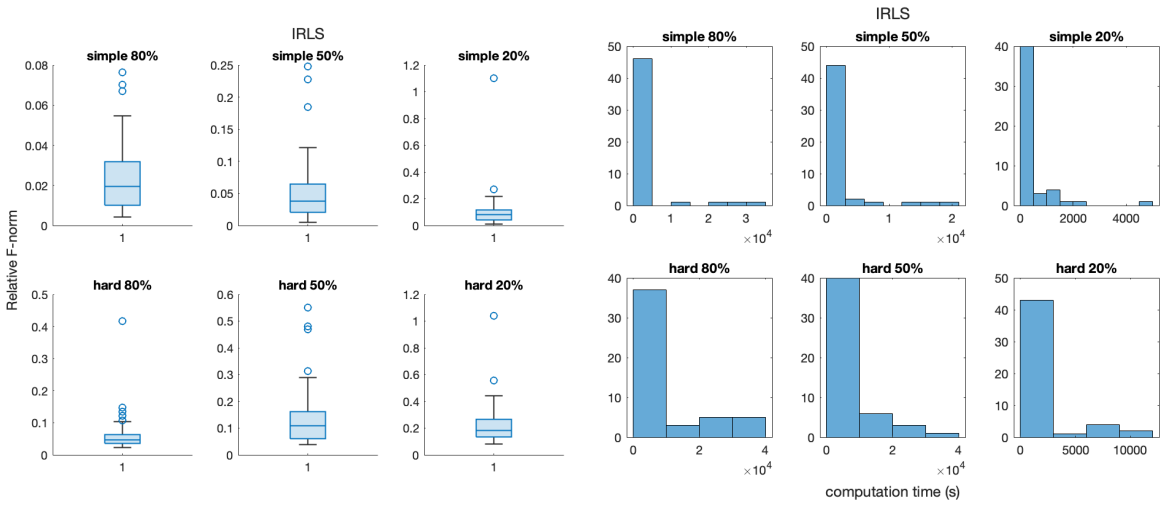
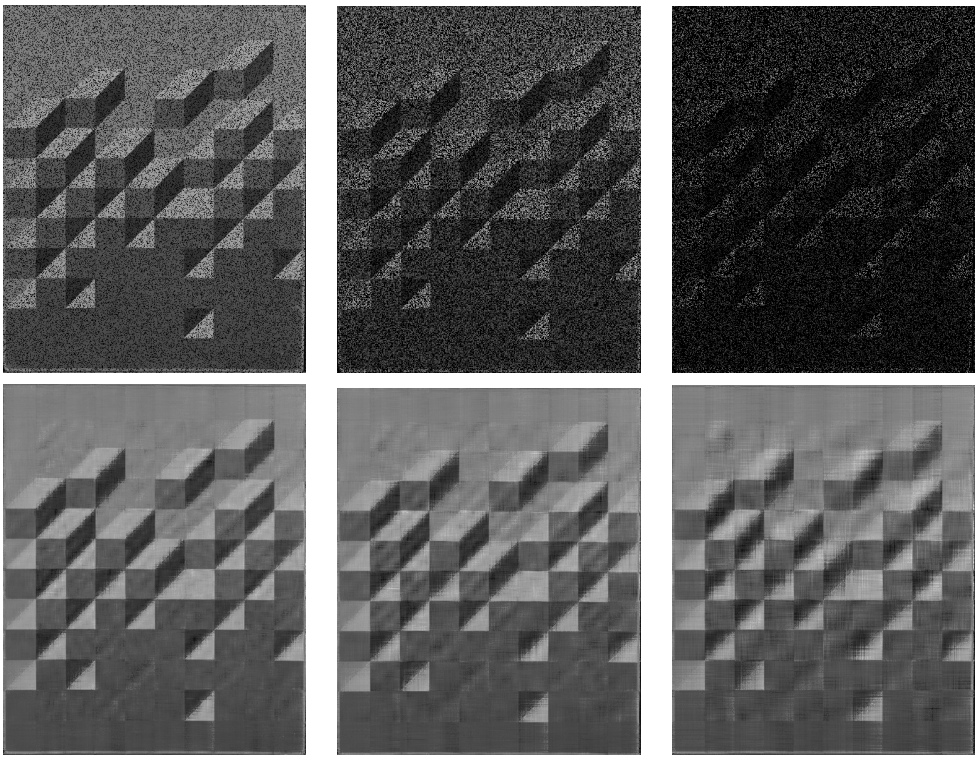


Figure 3.1: Relative residual Frobenius-norm (left) and run time (right) for IRLS algorithm. The upper row shows the result for simple test images and bottom row shows the result for hard test images. Each section has three columns. The left most column shows the result for images with 80% remaining pixels, the middle shows result with 50% remaining and the right shows result with 20% remaining.

The algorithm has two parts, the first part is to solve an optimization problem and the second part is to compute an SVD. Thus, the IRLS is much slower than SVT, and requires more memory. Similar to that of the SVT algorithm, there are ways to accelerate the computation for the SVD since only the first r singular values are needed, where r is the desired rank of the completed matrix.

3.4 Numerical Experiments



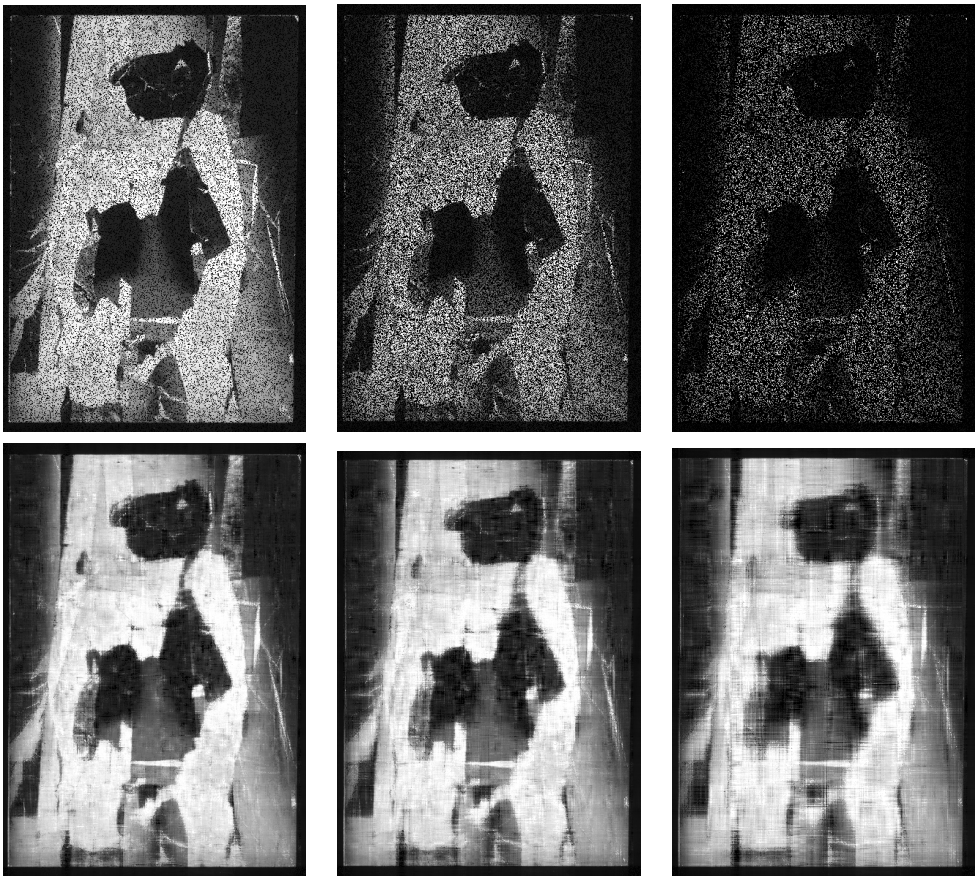
80% remaining 50% remaining 20% remaining

Figure 3.2: The original and recovered images of a simple problem with 80%, 50% and 20% remaining with the IRLS algorithm.

In this experiment, the IRLS algorithm requires an input rank. So to ensure the fairness across all numerical experiments, we used the rank output from the SVT algorithm to feed into the IRLS algorithm. With the same setup and running the same problem, the time for the IRLS algorithm is significantly longer. Especially for the hard problem, where some take hours to finish. On the other hand, the relative Frobenius-norm for the IRLS algorithm has some extreme values. Although the majority of the simple problems' relative residual Frobenius-norms fall within 0.2, in the case where only 20% of the original data are known, there are some values greater than 1, which is completely false. The relative residual Frobenius-norm error for a test case with only 20% known data without any process should be around

0.8. For an algorithm to get an error greater than 1 is worse than not applying any algorithm at all, so the error larger than 1 should not happen and it is false. The algorithm performs worse in dealing with hard and less structured problems. Even with 80% of the data, the algorithm still fails to converge, and the situation becomes worse as the amount of initial data decreases.

The rank for simple problems with 80%, 50%, 20% remaining data is 18, 12, 9; the rank for hard problems are 20, 13, 11 respectively. The simple problems work great for 80% data remaining but still lacks clarity for only 20% data and a low rank input. Maybe IRLS will perform better with better parameter tuning, but it will take significantly more time.



80% remaining 50% remaining 20% remaining

Figure 3.3: The original and recovered hard problem with 80%, 50% and 20% remaining with the IRLS algorithm.

Chapter 4

Low Rank Matrix Fitting

The problem of minimizing the nuclear norm (1.2) can have different formulations. For the problem scheme below, the previous SVT and IRLS algorithms can be effective. However, a common problem is the requirement of solving a threshold SVD sub-problem, which can cost a lot of time especially for a large scale problem.

So instead of solving the problem of minimizing nuclear norm, the authors in the paper [31] propose to factor matrix X into two smaller matrices J and K and then solve a minimize Frobenius-norm problem:

$$\begin{aligned} &\underset{J,K,Z}{\text{minimize}} \ \ \frac{1}{2}\|JK - Z\|_F^2, \text{ where } X = JK \\ &\text{subject to } \ \mathscr{P}_\Omega(Z) = \mathscr{P}_\Omega(M) \end{aligned} \tag{4.1}$$

where $X \in \mathbb{R}^{n_1 \times n_2}$ is the data matrix, $J \in \mathbb{R}^{n_1 \times r}$, $K \in \mathbb{R}^{r \times n_2}$ and $Z \in \mathbb{R}^{n_1 \times n_2}$. r should be the rank of data matrix X, but the rank is usually unknown so r is just a guessed rank just like the initial rank for the IRLS algorithm. Matrix Z at the first iteration is the data matrix, but it gets updated by matrix J and K in each iteration. The usual practice of factoring $X = JK$ is to first let K be the identity matrix and $J = ZK^T$

4.1 Iterative Steps

The iterative step to solve (4.1) is rather simple since it just needs to iteratively update J, K, Z respectively by solving an optimization problem in each iteration. For example, in iteration k, the scheme is:

$$\begin{aligned} J^k &= Z^{k-1}(K^{k-1})^\dagger = \text{argmin}_J \|J^{k-1}K^{k-1} - Z^{k-1}\|_F^2 \\ K^k &= (J^k)^\dagger Z^{k-1} = \text{argmin}_K \|J^k K^{k-1} - Z^{k-1}\|_F^2 \\ Z^k &= J^k K^k - \mathscr{P}_\Omega(M - J^k K^k) \end{aligned} \tag{4.2}$$

The dagger symbol $K^\dagger$ means the Moore-Penrose pseudo-inverse of matrix K, $K^\dagger = (K^T K)^{-1} K^T$. The last step is to enforce the known entries to match those in data matrix M. This scheme requires solving two optimization problems, which is not optimal for computation speed. Thus, an alternative scheme can be used, and the proof can be found in the original LMaFit paper [31]:

$$\begin{aligned} J^k &= Z^{k-1}K^{(k-1)\top} \\ K^k &= \text{argmin}_K \|J^k K^{k-1} - Z^{k-1}\|_F^2 \\ Z^k &= J^k K^k - \mathscr{P}_\Omega(M - J^k K^k) \end{aligned} \tag{4.3}$$

This scheme uses the orthogonal projection in Lemma 2.1 in the original paper [5], which states if J^k and K^k are generated by solving the two optimization problems (4.2), then $J^k K^k = Z^{k-1}(K^{k-1})^\top (K^{k-1}(Z^{k-1})^\top Z^{k-1}(K^{k-1})^\top)^\dagger (K^{k-1}(Z^{k-1})^\top) Z^{k-1}$.

Another alternative scheme also uses orthogonality and just simply computes $J^k K^k = VV^\top Z^{k-1}$ where V has columns of an orthogonal basis of $Z^{k-1}(K^{k-1})^\top$. This iterative scheme becomes extremely simple: just let $J^k = V$ and $K^k = V^\top Z^{k-1}$

$$\begin{aligned} [J^k, R] &= qr(Z^{k-1}(K^{k-1})^\top) \\ K^k &= J^{k\top} Z^{k-1} \\ Z^k &= J^k K^k - \mathscr{P}_\Omega(M - J^k K^k) \end{aligned} \tag{4.4}$$

Using this iterative scheme is very fast because the only significant cost is to compute the QR factorization of a matrix, which is generally much faster than computing an SVD factorization.

4.2 Algorithm

To sum up the previously discussed iterative steps, we obtain the following algorithm:

---

input : data matrix M, rank r, tolerance ϵ, size $[n_1, n_2] = size(M)$

output: completed matrix Z^k

---

initialization: stopping flag $flag = False$, counter $k = 0$, and

$Z^0 = M$,$J^0 = \texttt{zeros}(n_1, r)$, $K^0 = \texttt{eye}(r, n_2)$;

while *!flag* **do**

$k = k + 1$;

update J, K with the above iterative method 4.2, 4.3, 4.4;

$Z^k = J^k K^k - \mathcal{P}_\Omega(M - J^k K^k)$;

$flag = \frac{\|\mathcal{P}_\Omega(Z^k - M)\|_F}{\|\mathcal{P}_\Omega(M)\|_F} < \epsilon$;

end

Algorithm 3: LMaFit algorithm. In this algorithm, we use Matlab notation where `zeros` denotes a matrix of all zeros and `eye` is the identity matrix.

This algorithm is basically a three point iteration and only J, K, Z gets updated in each step.

4.3 Numerical Experiments

Using the same test images and running the QR factorization iteration scheme (4.4), the running time is extremely fast. The run time does not depend on the number

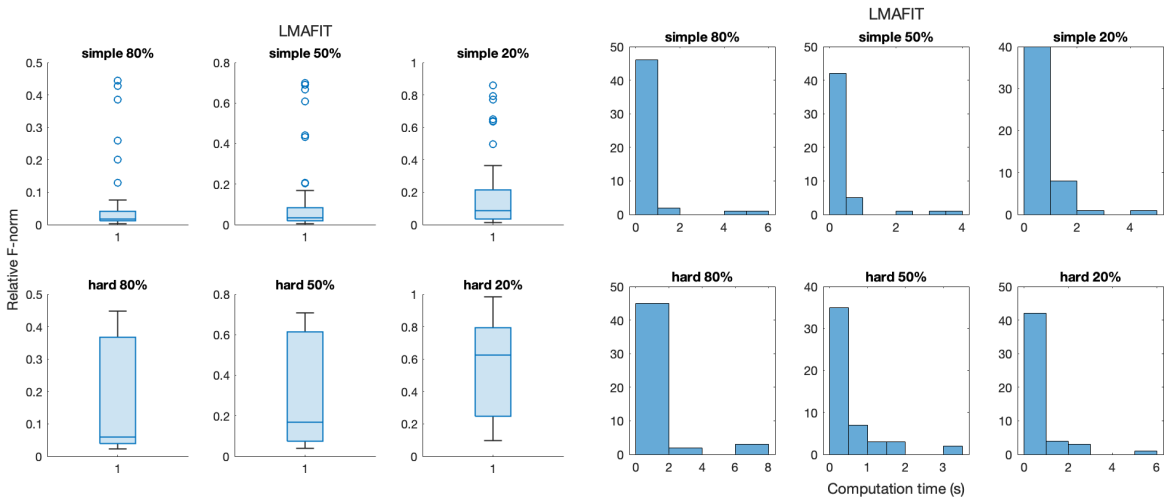
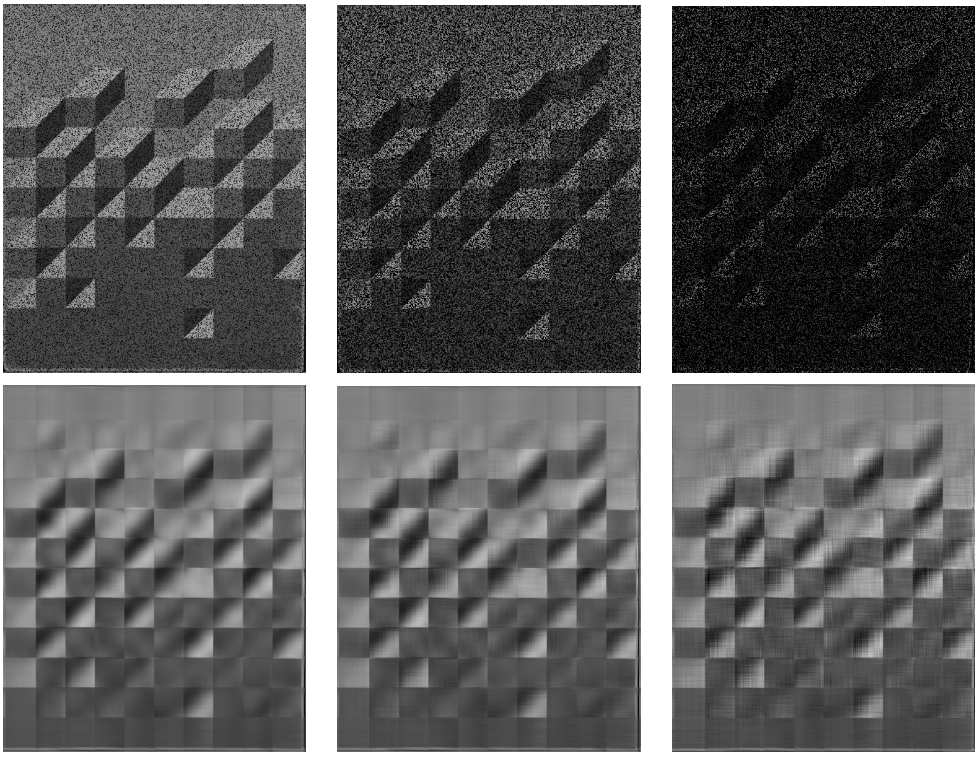


Figure 4.1: Relative Frobenius-norm (left) and run time (right) for LMaFit algorithm. The upper row shows the result for simple test images and bottom row shows the result for hard test images. Each section has three columns. The left most column shows the result for images with 80% remaining pixels, the middle shows result with 50% remaining and the right shows result with 20% remaining.

of data remaining, and it only depends on the size of the problem. However, the fast run time comes with some trade off: the relative Frobenius-norm is large. The result of using the QR scheme (4.4) or using optimization scheme (4.3) is very similar in terms of residual Frobenius-norm, so choosing a faster algorithm is optimal. For simple problems, the majority of results fall below 0.2, but the number of outliers increase as the input data decreases. The LMaFit algorithm works relatively well for simple problems, and it does not work well with hard problems. The third quartile for hard problems with 80% of remaining data is around 0.35, and the average for problems with 20% remaining data is even more than 0.5. From the below visual result, the imperfection of LMaFit's result can be seen more clearly.

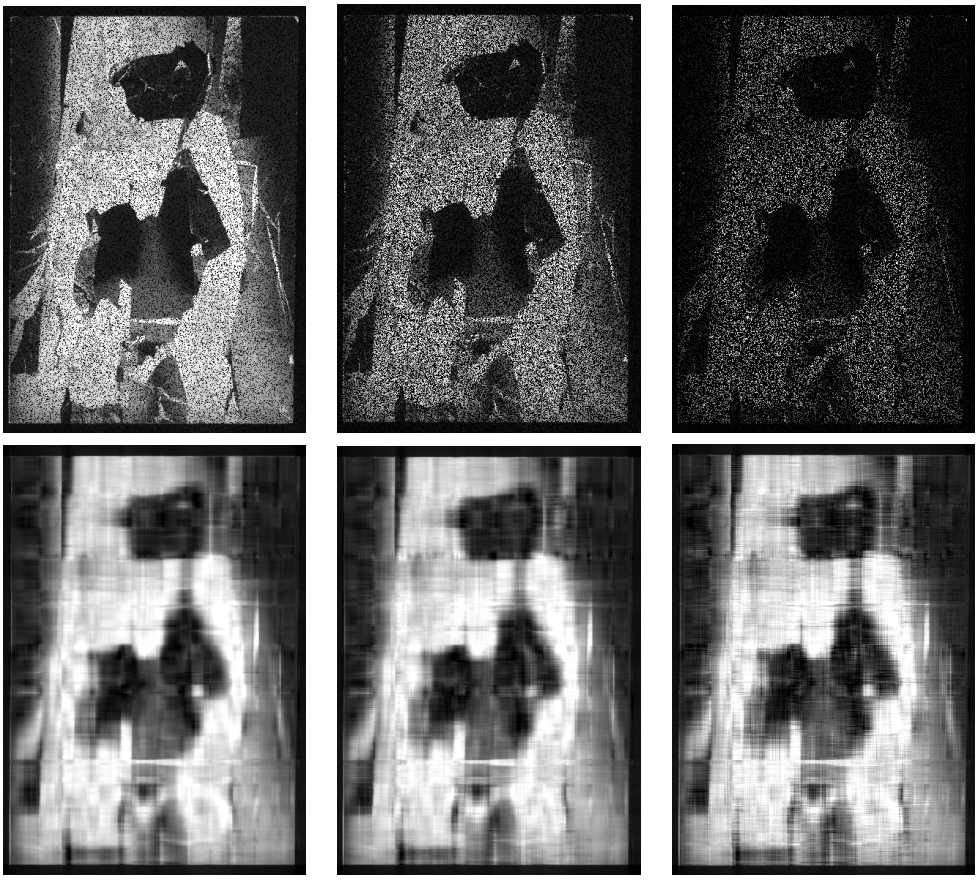


80% remaining 50% remaining 20% remaining

Figure 4.2: The original and recovered images of a simple problem with 80%, 50% and 20% remaining with LMaFit algorithm.

This experiment also uses the output rank from the SVT algorithm to feed in the parameter to decide the matrix factorization size. The recovered result for simple images only captures the general block structure and failed to illustrate the diagonal lines. Even for 80% of remaining data, the result even loses the input image's structure. The result of simple images shows a sign for overfitting because the bottom part of the image, which is suppose to be just a solid shape, has been recovered with a certain block structure.

The result for hard problems is even worse. The final image does not show any defined clarity of the original image's shape. So it is not hard to see why the relative Frobenius-norm is around 0.5 for hard problems.



80% remaining 50% remaining 20% remaining

Figure 4.3: The original and recovered images of a hard problem with 80%, 50% and 20% remaining with LMaFit algorithm.

Chapter 5

Comparison Across Algorithms

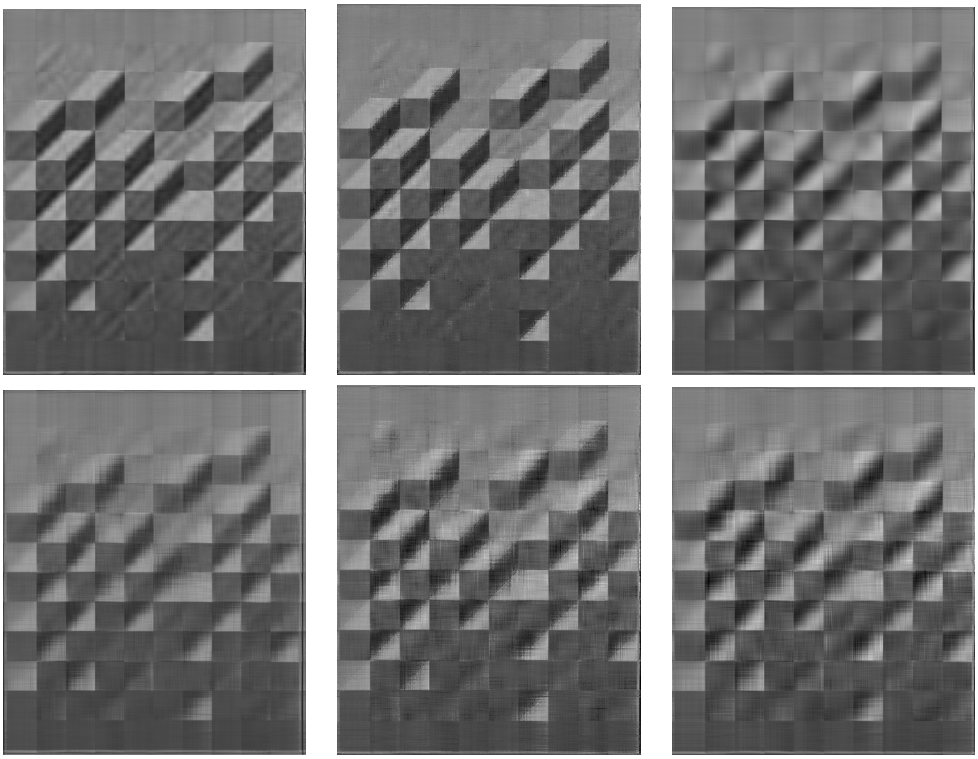


Figure 5.1: The recovered simple picture with 80% (top) and 20% (bottom) remaining data for all three algorithms.

Now, three algorithms for low rank matrix completion have been introduced: Singular Value Thresholding (Algorithm 1), Iteratively Reweighted Least Squares (Algorithm

2) and Low-rank Matrix Fit (Algorithm 3). In terms of computational speed, the LMaFit algorithm is the best because it only needs to compute a QR factorization. In terms of accuracy, the SVT and the IRLS algorithms are really similar to each other in terms of relative residual Frobenius-norm. IRLS has a slightly closer range of result and seems to be more stable in terms of convergence for a harder problem with less known data, but the problem for IRLS is the slow speed. The IRLS algorithm requires to solve a least squares problem and compute a singular value decomposition, which is slow. From Figure 5.2, the difference in recovered pictures for the simple problem with 80% and 20% of the original data can be easily observed.

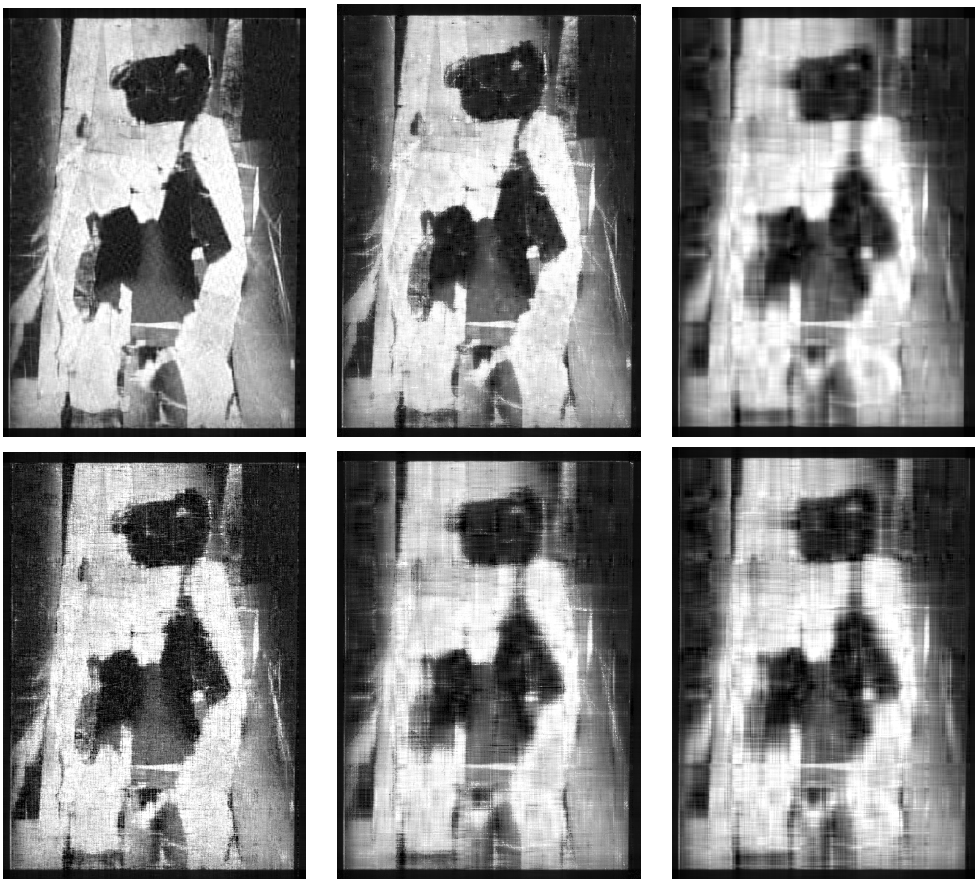


Figure 5.2: The recovered hard picture with 80% (top) and 20% (bottom) remaining data for all three algorithms

For easy problems in general, the IRLS algorithm gives a result with more clarity

especially in diagonal edges. Since the goal is low rank matrix completion, having a more defined diagonal edge with low rank constraint means the image is recovered well. In experiments with only 20% of original data, the IRLS and SVT algorithm shows a relatively defined diagonal edge, where the IRLS algorithm produces more contrast. The LMaFit algorithm does not produce a good result in terms of completing a diagonal shape. However, if the input rank for IRLS and LMaFit is higher, the result might be better.

For a hard problem, the low rank matrix completion algorithm is not designed to handle large unstructured data. These algorithms aim to complete new columns with existing columns and not to generate a new column. Thus, for hard problems, the recovered image has a certain vertical texture, which shows the effort of making the image low-rank. However, hard problems can still be used as a test for algorithms' performance. The SVT algorithm seems to have the best result for hard problems.

5.1 Problem of Negative Numbers

The above three algorithms do not have a constraint on negative values for the completed matrix. For SVT (Algorithm 1), the result from the SVD might contain negative values, and the same problem occurs with using linear solvers in the optimization problem in IRLS (Algorithm2) and LMaFit (Algorithm 3). In some cases of matrix completion, having negative values do not impose a problem. However, in the case of completing an image, having negative values in the result does not make sense since the pixel value should not be negative.

Thus, a solver that has nonnegative constraints can be incorporated when solving a least squares problem. IRtools [7] in MATLAB is a toolbox that can be used for this type of problem. The software package IR Tools provides implementations of a range of iterative solvers for large-scale ill-conditioned linear systems where regulariza-

tion is needed to stabilize computations. The solvers include iterative regularization methods where the regularization is due to the semi-convergence of the iterations, Tikhonov-type formulations where the regularization is explicitly formulated in the form of a regularization term, and methods that can impose bound constraints on computed solutions. The package also contains "hybrid" methods that use iterative Krylov subspace methods combined with SVD-based direct regularization methods on small subproblems at each iteration. The hybrid approach has the advantage that regularization parameters can be estimated at each iteration. The software package also contains a set of test problems that represent realistic large-scale problems found in image reconstruction and several other applications, but we use only the solvers. The solvers in the toolbox use the naming convention IRxxxx, where IR denotes "Iterative Regularization", and xxxx refers to a specific method. The methods we use in this thesis are:

IRfista: Fast Iterative Soft-Thresholding Algorithm, is an accelerated gradient descent like scheme.

IRmrnsd: Modified Residual Norm Steepest Descent, is a steepest descent scheme that constrains solutions to have nonnegative values.

IRnnfcgls: Nonnegative Flexible Conjugate Gradient Method for Least Squares Problems, is a Krylov subspace iterative method that constrains solutions to have nonnegative values.

All of the methods solve least squares problems, so the original Frobenius-norm must be transformed. Recall the original minimization problem is $K = \text{argmin}_K \|JK - Z\|_F^2$ with the second iterative scheme (4.3).

In order to use the IRtools package, we need to rewrite this minimization problem as a standard least squares matrix-vector formulation instead of the current matrix-matrix formulation. The trick here is to use Kronecker products. Specifically, if we write K and Z as

$$K = \begin{bmatrix} \boldsymbol{k_1} & \boldsymbol{k_2} & \cdots \boldsymbol{k_n} \end{bmatrix}, \quad Z = \begin{bmatrix} \boldsymbol{z_1} & \boldsymbol{z_2} & \cdots \boldsymbol{z_n} \end{bmatrix}$$

where $\boldsymbol{k_n}$ is the n^{th} column in matrix K and $\boldsymbol{z_n}$ is the n^{th} column in matrix Z. Then this problem satisfies:

$$J\boldsymbol{k_1} = \boldsymbol{z_1}$$
$$J\boldsymbol{k_2} = \boldsymbol{z_2}$$
$$\vdots$$
$$J\boldsymbol{k_n} = \boldsymbol{z_n}$$

Then the problem can be easily transformed into:

$$[J \otimes I] \begin{bmatrix} \boldsymbol{k_1} \\ \boldsymbol{k_2} \\ \vdots \\ \boldsymbol{k_n} \end{bmatrix} = \begin{bmatrix} J & 0 & \cdots & 0 \\ 0 & J & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & J \end{bmatrix} \begin{bmatrix} \boldsymbol{k_1} \\ \boldsymbol{k_2} \\ \vdots \\ \boldsymbol{k_n} \end{bmatrix} = \begin{bmatrix} J\boldsymbol{k_1} \\ J\boldsymbol{k_2} \\ \vdots \\ J\boldsymbol{k_n} \end{bmatrix} = \begin{bmatrix} \boldsymbol{z_1} \\ \boldsymbol{z_2} \\ \vdots \\ \boldsymbol{z_n} \end{bmatrix}$$

Thus, the least squares problem to solve is $\|[J \otimes I]\boldsymbol{k} - \boldsymbol{z}\|_2$ where $\boldsymbol{k}$ is result we want with K stretched to a vector by column, and $\boldsymbol{z}$ is a vector with Z stretched into a vector by column as well. The use of IRmrnsd, IRfista, and IRnnfcgls all show similiar residual Frobenius-norms. However, IRmrnsd has a relative lower running time compared with the other two methods, so the result of IRmrnsd will be discussed in this thesis. The modified LMaFit (Algorithm 4) just involves two extra steps of transforming the shape of the matrix. The use of Kronecker products can cost too much space for a large matrix, but the matrix for these experiments are all relative small matrices so space is not a problem. Moreover, the iterative methods can be implemented efficiently without explicitly forming any Kronecker products.

5.2 Algorithm

The LMaFit algorithm, using the previously discussed modifications, can be changed into the following new algorithm:

input : data matrix M, rank r, tolerance ϵ, size $[n_1, n_2] = size(M)$

output: completed matrix Z^k

initialization: stopping flag $flag = False$, counter $k = 0$, and

$Z^0 = M$,$J^0 = zeros(n_1, r)$, $K^0 = eye(r, n_2)$;

while *!flag* **do**

$k = k + 1$;

$J^k = Z^{k-1} K^{k-1\top}$;

$Z_T = \text{reshape}(Z^{k-1}, [n_1 \cdot n_2, 1])$;

$K_T = \text{IRmrnsd}(J \otimes I, Z_T)$;

$K^k = \text{reshape}(K_T, [r, n_2])$;

$Z^k = J^k K^k - \mathcal{P}_\Omega(M - J^k K^k)$;

$flag = \frac{\|\mathcal{P}_\Omega(Z^k - M)\|_F}{\|\mathcal{P}_\Omega(M)\|_F} < \epsilon$;

end

Algorithm 4: LMaFit Modified Algorithm

5.3 Modified LMaFit Result

From the plot of relative residual Frobenius-norm Figure 5.3, it is clear that the result improves than the original LMaFit algorithm using QR factorization.

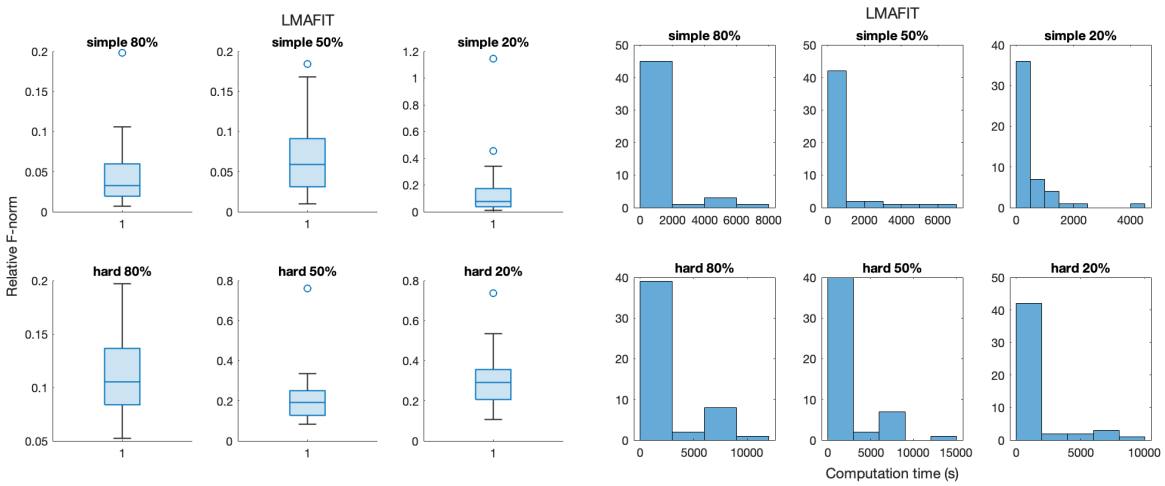
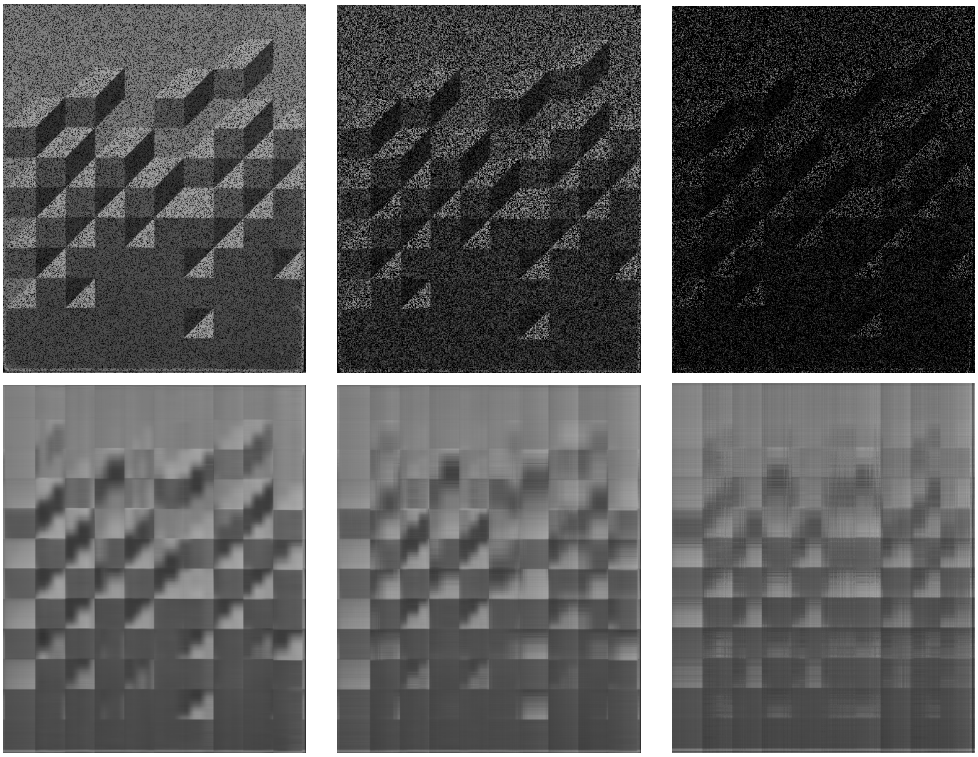


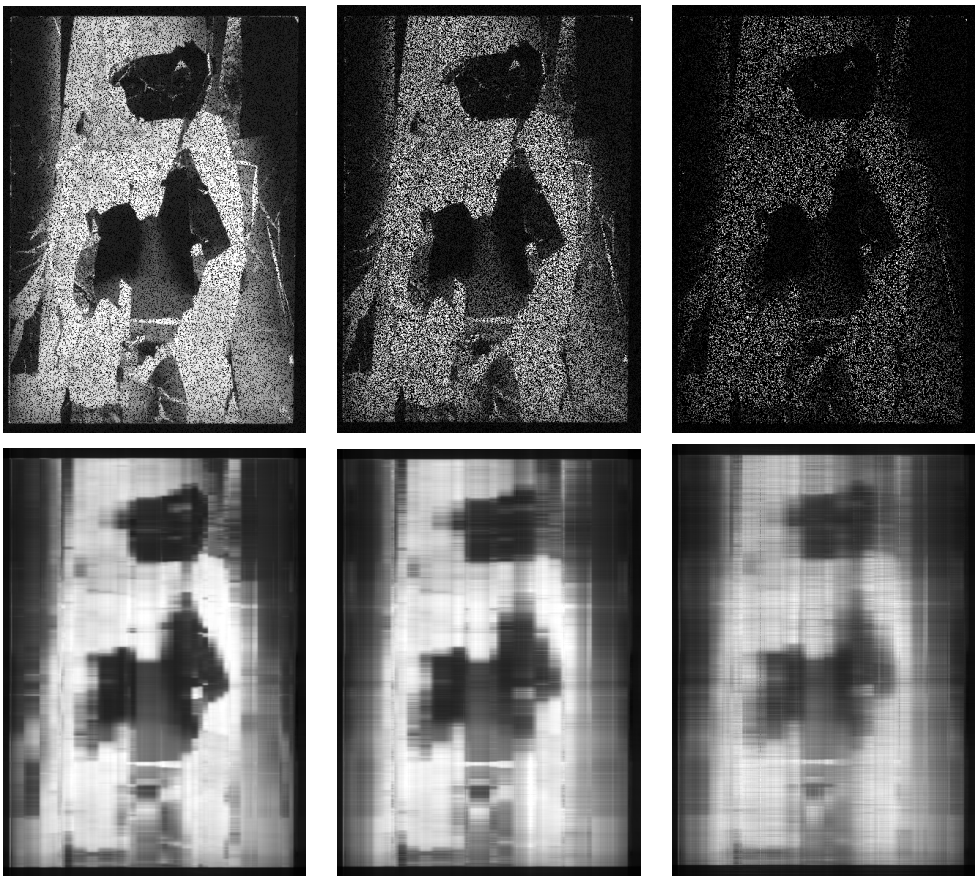
Figure 5.3: Relative Frobenius-norm (left) and run time (right) for the modified LMaFit algorithm. The upper row shows the result for simple test images and bottom row shows the result for hard test images. Each section has three columns. The left most column shows the result for images with 80% remaining pixels, the middle shows result with 50% remaining and the right shows result with 20% remaining.



80% remaining 50% remaining 20% remaining

Figure 5.4: The original and recovered simple problem with 80%, 50% and 20% remaining with the modified LMaFit algorithm.

However, just from looking at the recovered image, the modified algorithm does not seem to improve much compared with the original LMaFit algorithm. Especially for the hard problem, the recovered image from 20% original data left seems to have the same bad clarity than the image recovered from the original LMaFit algorithm. We make two remarks about this observation: firstly, this is only one of the images amongst all testing data, so the result varies across different problems; secondly, the modified algorithm has non-negative constraint, which improves the residual to some extent.



80% remaining 50% remaining 20% remaining

Figure 5.5: The original and recovered hard problem with 80%, 50% and 20% remaining with the modified LMaFit algorithm.

Chapter 6

Conclusions

This paper mainly discussed three low rank matrix completion algorithms: Singular Value Thresholding, Iteratively Reweighted Least Squares and Low-rank Matrix Fit. The test problems are abstract paintings from the MART dataset. All images were classified manually into simple and hard categories by the repetitiveness of the painting's structure. The result for the completed image is measured by the relative Frobenius-norm residual with the original image.

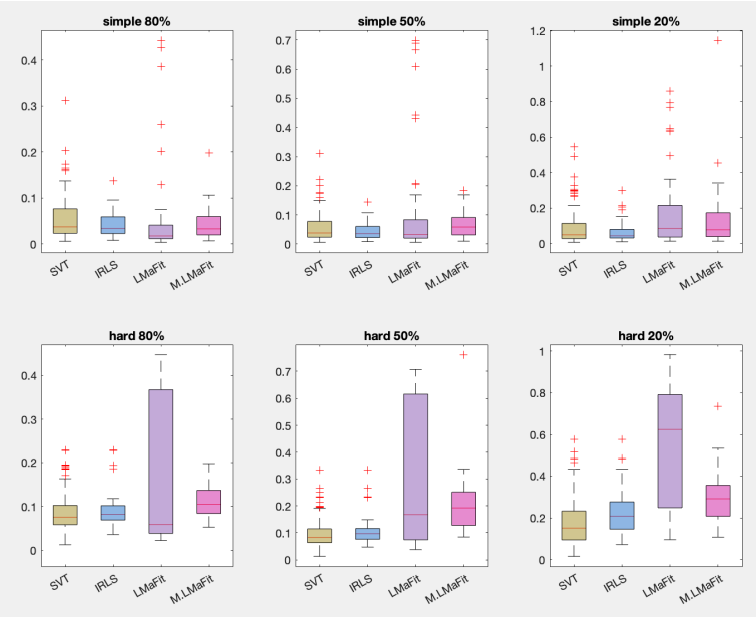


Figure 6.1: Relative Frobenius-norm for all four algorithms.

Amongst all four algorithms, total relative error shown in Figure 6.1 and total running time shown in Figure 6.2. the LMaFit has the fastest running speed but the least accurate result. The IRLS and SVT have similar results, but the SVT algorithm is significantly faster than the IRLS algorithm, although the result for simple problem is a little less accurate than the IRLS' result. Thus, in general the SVT algorithm is the algorithm of choice. The SVT algorithm has another advantage that there is no rank needed as an input parameter. In most cases, the rank of the original image is unknown, and a guess is often less optimal. Thus, with limited information, the SVT algorithm will give the best result with relatively fast time. However, all of the three algorithms do not have a non-negative constraint. Using the IRmrnsd method from IRtools [7] to modify the LMaFit algorithm yields better results than the original LMaFit algorithm. The benefit for this modified algorithm is the non-negative constraint, but the result is still not as good as the SVT or IRLS algorithm.

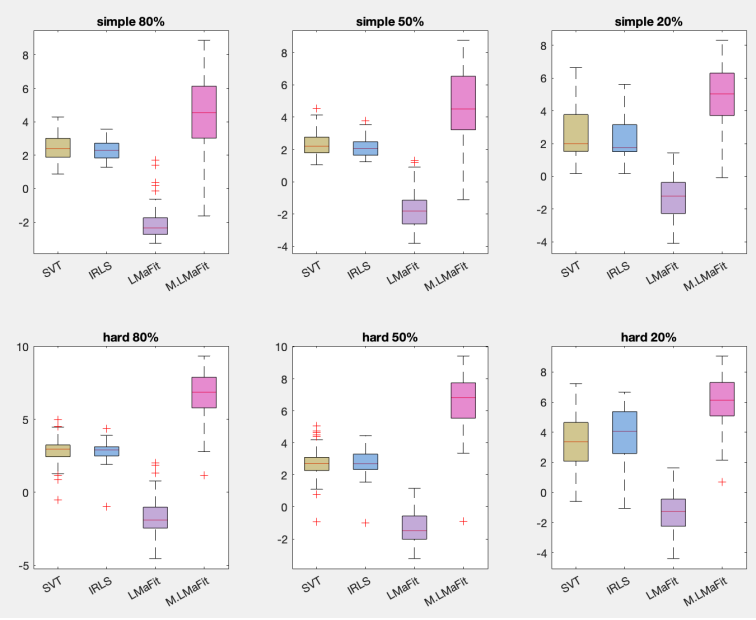


Figure 6.2: Relative running time for all four algorithms.

There are certain areas of improvements for this study as well. The first problem is assessing quality of the result. For a matrix, measuring with relative Frobenius-norm

residual is a valid option because it compares the value for each entry. However, in terms of measuring the quality of a recovered image, measuring pixel by pixel might not be a good choice. For example, an image shifted by 1 pixel compared to the original looks perfect but the relative Frobenius-norm residual may be large because the pixel values are not aligned. There are some existing methods, such as peak-signal-to-noise ratio (PSNR) or structural similarity index measure (SSIM) that measures the quality of an image in terms of the amount of noise and degradation [10]. A good recovered image has two aspects: a good quality and good similarity to the original image. Thus, a weighted average of image quality and relative error might be a better way to measure the result of the recovered image. Some previous studies on image processing uses area under receiver operating characteristic curve [9] and even can compare the similarity of features or structures in an image [34, 30].

Another area for this study to improve is to incorporate color. This thesis focuses on the gray-scale image for a faster computation, but the gray-scale image is derived from the three colored layers, red green and blue, of the original image. One way to incorporate color is to use low-rank matrix completion method for each colored layer, and then combine the layers together to form a color image. Another way is just to treat the whole color image as a 3D array or as a tensor and then apply tensor completion methods [17]. Some common methods for low-rank tensor completion includes tSVD (tensor Singular Value Decomposition) [35], tensor factorization [36] and Riemann optimization [14].

Bibliography

[1] Xavier Alameda-Pineda, Elisa Ricci, Yan Yan, and Nicu Sebe. Recognizing emotions from abstract paintings using non-linear matrix completion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.

[2] Jian-Feng Cai, Emmanuel J. Candes, and Zuowei Shen. A singular value thresholding algorithm for matrix completion, 2008.

[3] Jian-Feng Cai and Stanley Osher. Fast singular value thresholding without singular value decomposition. *Methods and Applications of Analysis*, 20(4):335–352, Dec 2013.

[4] Emmanuel J. Candes and Yaniv Plan. Matrix completion with noise. *Proceedings of the IEEE*, 98(6):925–936, 2010.

[5] Massimo Fornasier, Holger Rauhut, and Rachel Ward. Low-rank matrix recovery via iteratively reweighted least squares minimization, 2011.

[6] Matan Gavish and David L. Donoho. The optimal hard threshold for singular values is $4/sqrt(3)$, 2014.

[7] Silvia Gazzola, Per C. Hansen, and James G. Nagy. IR Tools: a MATLAB package of iterative regularization methods and large-scale test problems. *Numer Algor*, 81:773–811, 2019.

[8] Ioannis Giakoumis and Ioannis Pitas. Digital restoration of painting cracks. In *ISCAS '98. Proceedings of the 1998 IEEE International Symposium on Circuits and Systems (Cat. No.98CH36187)*, volume 4, pages 269–272 vol.4, 1998.

[9] Kenneth M. Hanson. Method of evaluating image-recovery algorithms based on task performance. *J. Opt. Soc. Am. A*, 7(7):1294–1304, Jul 1990.

[10] Alain Horé and Djemel Ziou. Image quality metrics: PSNR vs. SSIM. In *2010 20th International Conference on Pattern Recognition*, pages 2366–2369, 2010.

[11] Yao Hu, Debing Zhang, Jun Liu, Jieping Ye, and Xiaofei He. Accelerated singular value thresholding for matrix completion. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, page 298–306, New York, NY, USA, 2012. Association for Computing Machinery.

[12] Baohua Huang and Changfeng Ma. An iterative algorithm for the least Frobenius norm least squares solution of a class of generalized coupled Sylvester-transpose linear matrix equations. *Applied Mathematics and Computation*, 328:58–74, 2018.

[13] Hui Ji, Chaoqiang Liu, Zuowei Shen, and Yuhong Xu. Robust video denoising using low rank matrix completion. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1791–1798, 2010.

[14] Hiroyuki Kasai and Bamdev Mishra. Low-rank tensor completion: a Riemannian manifold preconditioning approach, 2016.

[15] Oleg Kuybeda, Gabriel A. Frank, Alberto Bartesaghi, Mario Borgnia, Sriram Subramaniam, and Guillermo Sapiro. A collaborative framework for 3D alignment and classification of heterogeneous subvolumes in cryo-electron tomography. *Journal of Structural Biology*, 181(2):116–127, Feb 2013.

[16] Rasmus M. Larsen. Lanczos bidiagonalization with partial reorthogonalization. (DAIMI PB-357), 1998.

[17] Ji Liu, Przemyslaw Musialski, Peter Wonka, and Jieping Ye. Tensor completion for estimating missing values in visual data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1):208–220, 2013.

[18] Canyi Lu, Changbo Zhu, Chunyan Xu, Shuicheng Yan, and Zhouchen Lin. Generalized singular value thresholding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 29(1), Feb. 2015.

[19] Si Lu, Xiaofeng Ren, and Feng Liu. Depth enhancement via low-rank matrix completion. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3390–3397, 2014.

[20] Xiaoqiang Lu, Tieliang Gong, Pingkun Yan, Yuan Yuan, and Xuelong Li. Robust alternative minimization for matrix completion. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(3):939–949, 2012.

[21] Saeid Mehrdad and Mohammad H. Kahaei. Deep learning approach for matrix completion using manifold learning. *Signal Processing*, 188:108231, Nov 2021.

[22] Ryan J. Meuth, Paul Robinette, and Donald C. Wunsch. Computational intelligence meets the netflix prize. In *2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence)*, pages 686–691, 2008.

[23] Luong Trung Nguyen, Junhan Kim, and Byonghyo Shim. Low-rank matrix completion: A contemporary survey. *IEEE Access*, 7:94215–94237, 2019.

[24] Nikos Nikolaidis and Ioannis Pitas. Digital image processing in painting restora-

tion and archiving. In *Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205)*, volume 1, pages 586–589 vol.1, 2001.

[25] Pulak Purkait and Bhabatosh Chanda. Digital restoration of damaged mural images. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing*, ICVGIP '12, New York, NY, USA, 2012. Association for Computing Machinery.

[26] Benjamin Recht, Maryam Fazel, and Pablo A. Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM Review*, 52(3):471–501, Jan 2010.

[27] Günther Reitberger and Tomas Sauer. Background subtraction using adaptive singular value decomposition. *Journal of Mathematical Imaging and Vision*, 62, 10 2020.

[28] Gábor Takács, István Pilászy, Bottyán Németh, and Domonkos Tikk. Matrix factorization and neighbor based algorithms for the Netflix prize problem. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, RecSys '08, page 267–274, New York, NY, USA, 2008. Association for Computing Machinery.

[29] Hongxing Wang. On least squares solutions subject to a rank restriction. *Linear and Multilinear Algebra*, 63(2):264–273, 2015.

[30] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612, 2004.

[31] Zaiwen Wen, Wotao Yin, and Yin Zhang. Solving a low-rank factorization model for matrix completion by a nonlinear successive over-relaxation algorithm. *Math Program Comp*, 4:333–361, 2012.

[32] Hongyang Xue, Shengming Zhang, and Deng Cai. Depth image inpainting: Improving low rank matrix completion with low gradient regularization. *IEEE Transactions on Image Processing*, 26(9):4311–4320, 2017.

[33] Victoria Yanulevskaya, Jasper Uijlings, Elia Bruni, Andreza Sartori, Elisa Zamboni, Francesca Bacci, David Melcher, and Nicu Sebe. In the eye of the beholder: Employing statistical analysis and eye tracking for analyzing abstract paintings. In *Proceedings of the 20th ACM International Conference on Multimedia*, MM '12, page 349–358, New York, NY, USA, 2012. Association for Computing Machinery.

[34] Lin Zhang, Lei Zhang, Xuanqin Mou, and David Zhang. FSIM: A feature similarity index for image quality assessment. *IEEE Transactions on Image Processing*, 20(8):2378–2386, 2011.

[35] Zemin Zhang and Shuchin Aeron. Exact tensor completion using t-SVD, year=2017, volume=65, number=6, pages=1511-1526, doi=10.1109/TSP.2016.2639466. *IEEE Transactions on Signal Processing.*

[36] Pan Zhou, Canyi Lu, Zhouchen Lin, and Chao Zhang. Tensor factorization for low-rank tensor completion. *IEEE Transactions on Image Processing*, 27(3):1152–1163, 2018.