The Application of Matrix Completion and PCA in the Movie Recommendation System

Jialin Wang

August 15, 2022

Abstract

The DURF project mainly does a theoretical analysis of Matrix Completion using the Singular Value Thresholding Algorithm and the Robust Principal Component Analysis with different algorithms, and how Sparse PCA can be applied to analyze the problem. We propose two general methods to deal with sparse matrix completion problems. The former matrix completion method being researched is to approximate the sparse data matrix with minimum nuclear norm following specific convex constraints. By means of an iterative soft-thresholding operation on singular values, we can complete the matrix without changing the matrix size or form. Moreover, researches will also be done concerning the principal component analysis, where we could explore the principal components of the sparse matrix, and thus it is possible to recover the matrix with missing entries to a full matrix. Following this logic, due to the limitation of classical PCA, we will focus on Robust Principal Component Analysis to solve the matrix completion problem. The algorithms being implemented to RPCA are the Augmented Lagrange Multiplier (ALM) Method and the Dual Method. We will carry out theoretical analysis and real-data experiment performance comparison of the three algorithms mentioned all above. Apart from that, we would try to figure out how Sparse PCA could be applied to analyze the problem. Finally, an experiment on the real data implementation will be realized to demonstrate the feasibility of the above-mentioned algorithms, and the real data comes from both the famous Netflix movie challenges as well as real-life settings.

Keywords— Matrix completion, nuclear norm minimization, singular value thresholding, robust PCA, sparse PCA

1 Introduction

1.1 Matrix Completion

Matrix completion is to impute and fill in missing entries in an old matrix, and the method is widely applied in many data settings. One of the most famous example is the Netflix movie rating problem and in the challenge, competitors tried to improve the recommendation system. Since the original data matrix is very sparse and only 1% of the ratings are present, they needed to invent algorithms to complete the missing entries in order to predict unrated

movies' ratings for customers.

The common way to deal with the matrix completion problem is to impose a rank constraint on the sparse matrix. Given a data matrix $\mathbf{Z}$ whose entries are indexed by the subset $\Omega \subset \{1, ..., m\} \times \{1, ..., n\}$, and Ω is a random subset of cardinality s, we thus derive the natural approach to find the lowest rank approximating matrix $\hat{\mathbf{Z}}$. The matrix $\hat{\mathbf{Z}}$ interpolates the observed entries of data matrix $\mathbf{Z}$, and the expression is:

$$\text{minimize} \;\; \text{rank}(\mathbf{M}) \;\; \text{subject to} \;\; m_{ij} = z_{ij} \;\; \text{for all} \;\; (i,j) \in \Omega \tag{1}$$

Although it is not as ill-posed as people thought, the optimization problem is actually computationally intractable (NP-hard), and is not applicable for even moderately large matrices, not to mention the Netflix problem whose dataset has 17770 columns (movies) and 48189 rows (customers).

The nuclear norm is the sum of the singular values of a matrix, which is also called the trace norm. Candès and Recht's result proves that under suitable conditions, the rank minimization program (1) is formally equivalent to the following program because they have the same unique solution [3]:

$$\text{minimize} \;\; \|\mathbf{M}\|_* \;\; \text{subject to} \;\; m_{ij} = z_{ij} \;\; \text{for all} \;\; (i,j) \in \Omega \tag{2}$$

The rationale behind the matrix completion using minimum nuclear norm rather than rank minimization is that the problem turns into a convex one. To be specific, with nuclear norm - the sum of singular values, it turns into a semi-definite program, which belongs to the convex programs and special solvers can be applied to find the solution. What is more, sometimes the observed entries are with noise, so we need to tolerate noise and we have the following relaxed version of the program, which is called spectral regularization:

$$\text{minimize}_{\mathbf{M}} \;\; \frac{1}{2}\sum (z_{ij} - m_{ij})^2 + \lambda \|\mathbf{M}\|_* \tag{3}$$

This program reduces the potential overfitting in the case of noisy entries. λ is a hyperparameter that must be chosen from cross-validation. Typically, we do not necessarily require the first term to be zero, otherwise, λ will be sufficiently small and has no function of penalty.

1.2 Principal Component Analysis

Principal Component Analysis (PCA) is a dimensionality-reduction method that is often used to reduce the dimensionality of large datasets. Usually, we apply singular value decomposition to the centered data matrix and obtain the top-k left singular vectors, which span the projection of the subspace approximating the centered data matrix. From another perspective, the principal components are the directions where the points spread out the most and explain as much of the variance in the data as possible. However, when it comes to a sparse or corrupted data matrix, we need to introduce Sparse Principal Component Analysis (sparse PCA) and Robust Principal Component Analysis (RPCA). Theoretically speaking, ordinary PCA is going to break down very badly if the number of variables is larger than the number of sample size (Johnstone 2001). In other words, classical PCA is brittle regarding grossly corrupted or outlying observations [5], which is very common in modern applications. Ordinary PCA is very sensitive to large errors. For any symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ with normalized

eigenvectors $\mathbf{u}_1, \mathbf{u}_2, ..., \mathbf{u}_n$, the i-th principal component can be expressed as:

$$\mathbf{u}_i = \text{argmax}_{\|\mathbf{v}\|=1, \mathbf{v} \perp \mathbf{u}_1, ..., \mathbf{u}_{i-1}} \mathbf{v}^T \mathbf{A} \mathbf{v} \tag{4}$$

From another perspective, for a ground truth data matrix M which could be decomposed as $M = L_0 + N_0$, where L_0 is low-rank and N_0 is a small perturbation matrix, the classical PCA finds the best rank-k estimate of L_0 by dealing with [2]:

$$\text{minimize} \quad \|M - L\| \quad \text{subject to} \quad \text{rank(L)} \leq k \tag{5}$$

This program could be solved by singular value decomposition and has abundant properties when the noise N_0 is small and independent and identically distributed Gaussian.

1.3 Sparse Principal Component Analysis

How to combine the maximum variance characterization of PCA with sparsity? For sparse PCA, the intention is to explain as much variability in the data as possible, and at the same time, use principal components extracted from as few as variables as possible [7]. Thus, it will enhance and facilitate the interpretation of the variables and model. In order to realize this purpose, many approaches are brought about. For example, the famous SCoTLASS procedure of Jolliffe, Trendafilov and Uddin [6] imposes the L1-norm restriction onto the criterion, leading to the following program:

$$\text{maximize}_{\|v\|_2=1} \ \{v^T \mathbf{X}^T \mathbf{X} v\} \quad \text{subject to} \ \|v\|_1 \leq t \tag{6}$$

The L1-norm restriction makes some of the loadings to be zero, which the v is sparse. Though the L1-norm makes a convex problem, the whole problem remains nonconvex, and is not suitable for a simple iterative algorithms.

In order to fix the problem of non-convexity, we propose a semi-definite program to realize the Sparse PCA [4]. And we will discuss this topic in detail and try to explain the concept with decomposition algorithms in the later section.

1.4 Robust Principal Component Analysis

Since performance and applicability of the classical PCA in real applications are constraint by a lack of robustness to corrupted observations or outlying, recent advances in sparsity and robust statistics enable us to recover the corrupted and sparse data matrix, and several potential approaches to robustifying PCA have been brought up and put in the limelight. One proposal, which is called Robust Principal Component Analysis [2], is that we recover a low-rank matrix $L :_0$ from corrupted observations $M = L_0 + S_0$, where M is the ground truth, and is decomposed as the low-rank matrix L_0 and the sparse matrix S_0. Unlike the small noise term N_0 in the classical PCA, the entries of S_0 could have arbitrarily large magnitude, and their support is assumed to be sparse but unknown. As a result, following the same manner as classical PCA, the program we need to solve is:

$$\text{min}_{L,S} \ \ \text{rank(L)} + \|S\|_0 \quad \text{subject to} \ \ L + S = M \tag{7}$$

However, the program is strictly non-convex due to the rank and zero norm, so we cannot solve the problem in a computationally efficient way. They [2] introduce proxies for these

norms and turn it into a tractable convex optimization:

$$\text{minimize} \quad \|L\|_* + \lambda\|S\|_1 \quad \text{subject to} \quad L + S = M \tag{8}$$

where $\|L\|_* := \sum_i \sigma_i(L)$ denotes the nuclear norm of the matrix L (the sum of the singular values of M), and $\|S\|_1 = \sum_i j|S_{ij}|$ denotes the L1-norm of S seen as a vector in $\mathbb{R}^{m\times n}$. This problem recovers the low-rank L_0 and the sparse S_0. Since it is a convex relaxation, then, we will explore some efficient and scalable algorithms to solve the above program.

2 Algorithms

We will introduce three algorithms, the singular values thresholding algorithm for the nuclear norm minimization program, and the augmented Lagrange multiplier method and dual method for the RPCA matrix completion problem.

2.1 The Singular Value Thresholding Algorithm [1]

2.1.1 Algorithm Outline

Since minimizing the nuclear norm is a productive way to complete sparse matrices and recovers a low-rank matrix subject to restrictions, it is of great interest to develop an algorithm to solve the program (2). A numerical method being researched is the singular value thresholding algorithm, which deals with nuclear norm minimization problem of the following form:

$$\text{minimize} \quad \|\mathbf{X}\|_* \quad \text{subject to} \quad \mathcal{A}(\mathbf{X}) = \mathbf{b} \tag{9}$$

where $\mathcal{A}$ is a linear operator acting on matrices of size $m \times n$ and $\mathbf{b} \in \mathbb{R}^s$. This method applies to matrices of large size and will produce solutions with low rank. To fit for the sparse matrix completion setting, we introduce the projection operator $\mathcal{P}_\Omega : \mathbb{R}^{m\times n} \mapsto \mathbb{R}^{m\times n}$ as follows:

$$[\mathcal{P}_\Omega(\mathbf{Z})]_{ij} = \begin{cases} z_{ij} & \text{if } (i,j) \in \Omega \\ 0 & \text{if } (i,j) \in \Omega \end{cases} \tag{10}$$

As a result, our program can be expressed as:

$$\text{minimize} \quad \|\mathbf{X}\|_* \quad \text{subject to} \quad \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}) \tag{11}$$

where $\mathbf{X} \in \mathbb{R}^{m\times n}$ is an optimization variable. We fix $\tau \geq 0$ and a sequence $\{\delta_k\}_{k\geq 1}$ of scalar step sizes. Then we start with $\mathbf{Y}^0 = 0 \in \mathbb{R}^{m\times n}$, and iteratively the algorithm defines [1]:

$$\begin{cases} \mathbf{X}^k = \text{shrink}(\mathbf{Y}^{k-1}, \tau), \\ \mathbf{Y}^k = \mathbf{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\mathbf{M} - \mathbf{X}^k) \end{cases} \tag{12}$$

And $\text{shrink}(\mathbf{Y}, \tau)$ is a nonlinear function applying a soft-thresholding rule at level τ to the singular values of the input matrix. Procedures for deriving the program will be discussed in detail later. Here, according to Cai, Candès and Shen, there are two vital remarks of this program worth mentioning:

1. *Sparsity.* For $k \geq 0$, $\mathbf{Y}^k$ vanishes outside of Ω and is sparse. As a result, the shrink function can be evaluated quickly.

2. *Low-rank property.* The matrix $\mathbf{X}^k$ has low rank and therefore, the algorithm only requires a minimum storage to save the principal components in memory. Thus, the method is powerful in solving large size matrix completion problem.

2.1.2 Algorithm Detail Analysis

The definition of the singular value decomposition (SVD) of a matrix $\mathbf{X} \in \mathbb{R}^{m\times n}$ of rank r is:

$$\mathbf{X} = \mathbf{U}\Sigma\mathbf{V}^T, \quad \Sigma = \text{diag}(\{\sigma_i\}_{1\le i\le r}) \tag{13}$$

And here $\mathbf{U}$ is a matrix of size $m \times r$ and $\mathbf{V}$ is of size $n \times r$. They both have orthogonal columns and are called left and right singular vectors respectively. Additionally, the singular values σ_i are positive, whose sum is the nuclear norm mentioned before. For each $\tau \ge 0$, we define the soft-thresholding operator $\mathcal{D}_\tau$ as:

$$\mathcal{D}_\tau(\mathbf{X}) := \mathbf{U}\mathcal{D}_\tau(\Sigma)\mathbf{V}^T, \quad \mathcal{D}_\tau(\Sigma) = \text{diag}(\{(\sigma_i - \tau)_+\}), \tag{14}$$

where $p_+ = \max(0, p)$. In other words, this operator imposes a soft-thresholding constraint to the singular values of matrix $\mathbf{X}$, which effectively shrinks these towards zero. When some of the singular values of $\mathbf{X}$ are smaller than the threshold τ, many turning to zero, the rank of $\mathcal{D}_\tau(\mathbf{X})$ may become lower than that of $\mathbf{X}$, leading to sparse outputs [1]. Additionally, for each $\tau \ge 0$ and $\mathbf{Y} \in \mathbb{R}^{m\times n}$, the singular value shrinkage operator follows:

$$\mathcal{D}_\tau(\mathbf{Y}) = \arg\min_{\mathbf{X}}\{\frac{1}{2}\|\mathbf{X} - \mathbf{Y}\|_F^2 + \tau\|\mathbf{X}\|_*\} \tag{15}$$

As a result, the original program (11) can be written as:

$$\text{minimize} \;\; \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2 \quad \text{subject to} \;\; \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}) \tag{16}$$

The next thing is to derive the iteration (12). Here, we introduce the Lagrange multiplier method. Firstly, we set $f_\tau(\mathbf{X}) = \tau\|\mathbf{X}\|_* + \frac{1}{2}\|\mathbf{X}\|_F^2$ for some fixed $\tau > 0$, thus the original problem turns into:

$$\text{minimize} \;\; f_\tau(\mathbf{X}) \quad \text{subject to} \;\; \mathcal{P}_\Omega(\mathbf{X}) = \mathcal{P}_\Omega(\mathbf{M}) \tag{17}$$

The Lagrangian for this program is given by $\mathcal{L}(\mathbf{X}, \mathbf{Y}) = f_\tau(\mathbf{X}) + \langle \mathbf{Y}, \mathcal{P}_\Omega(\mathbf{M} - \mathbf{X})\rangle$, where $\mathbf{Y} \in \mathbb{R}^{m\times n}$. Strong duality holds and $\mathbf{X}^*$ and $\mathbf{Y}^*$ are primal-dual optimal, obeying

$$\sup_{\mathbf{Y}}\inf_{\mathbf{X}}\mathcal{L}(\mathbf{X}, \mathbf{Y}) = \mathcal{L}(\mathbf{X}^*, \mathbf{Y}^*) = \inf_{\mathbf{X}}\sup_{\mathbf{Y}}\mathcal{L}(\mathbf{X}, \mathbf{Y}) \tag{18}$$

(The function $g_0(\mathbf{Y}) = \inf_{\mathbf{X}}\mathcal{L}(\mathbf{X}, \mathbf{Y})$ is called the dual function.) Then we apply Uzawa's algorithm to find a saddle point of the problem, to obtain an iterative procedure. We inductively define from $\mathbf{Y}_0 = 0$ that:

$$\begin{cases} \mathcal{L}(\mathbf{X}^k, \mathbf{Y}^{k-1}) = \min_{\mathbf{X}}\mathcal{L}(\mathbf{X}, \mathbf{Y}^{k-1}) \\ \mathbf{Y}^k = \mathbf{Y}^{k-1} + \delta_k\mathcal{P}_\Omega(\mathbf{M} - \mathbf{X}^k), \end{cases} \tag{19}$$

where $\{\delta_k\}_{k\ge1}$ is a sequence of positive scalar step sizes. Uzawa's algorithm is indeed a subgradient method applied to the dual problem and each step moves the iteration in the direction of the gradient or a subgradient. We observe that $\partial_{\mathbf{Y}} g_0(\mathbf{Y}) = \partial_{\mathbf{Y}}\mathcal{L}(\tilde{\mathbf{X}}, \mathbf{Y}) = \mathcal{P}_\Omega(\mathbf{M} - \tilde{\mathbf{X}})$ where $\tilde{\mathbf{X}}$ is the minimizer of the Lagrangian for the value of $\mathbf{Y}$. As a result, the gradient

descent update for $\mathbf{Y}$ is of the form:

$$\mathbf{Y}^k = \mathbf{Y}^{k-1} + \delta_k \partial_{\mathbf{Y}} g_0(\mathbf{Y}^{k-1}) = \mathbf{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\mathbf{M} - \mathbf{X}^k) \tag{20}$$

Apart from that, we need to compute the minimizer of the Lagrangian (19), and note that:

$$\arg\min f_\tau(\mathbf{X}) + \langle \mathbf{Y}, \mathcal{P}_\Omega(\mathbf{M} - \mathbf{X}) \rangle = \arg\min \{ \frac{1}{2} \|\mathbf{X} - \mathcal{P}_\Omega \mathbf{Y}\|_F^2 + \tau \|\mathbf{X}\|_* \} \tag{21}$$

And from (15) we know that the minimizer is given by $\mathcal{D}_\tau(\mathcal{P}_\Omega(\mathbf{Y}))$ and also $\mathbf{Y}^k = \mathcal{P}_\Omega(\mathbf{Y}^k)$ for all $k \geq 0$, Uzawa's algorithm takes the form:

$$\begin{cases} \mathbf{X}^k = \mathcal{D}_\tau(\mathbf{Y}^{k-1}, \tau), \\ \mathbf{Y}^k = \mathbf{Y}^{k-1} + \delta_k \mathcal{P}_\Omega(\mathbf{M} - \mathbf{X}^k) \end{cases} \tag{22}$$

This is exactly the update of (12) and we finish the derivation. For the program (22), the shrinkage iteration is very simple to implement. At each iteration, we only need to compute an SVD and perform elementary matrix operations. The program could be coded easily with the help of a standard numerical linear algebra package. Real implementation will be discussed in later chapters.

2.2 Augmented Lagrange Multiplier Method [8]

For the RPCA problem (8), we apply the augmented Lagrange multiplier method for matrix recovery problem by identifying:

$$X = (L, S), \quad f(X) = \|L\|_* + \lambda \|S\|_1, \quad \text{and } h(X) = M - L - S \tag{23}$$

Then the Lagrangian function is:

$$L(L, S, Y, \mu) = \|L\|_* + \lambda \|S\|_1 + \langle Y, M - L - S \rangle + \frac{\mu}{2} \|M - L - S\|_F^2 \tag{24}$$

To solve this program, we introduce RPCA via the Exact ALM Method as well as as the Inexact ALM Method. We will not discuss the content and derivation of these algorithms in detail, but it is noting that the Inexact ALM Method is more quickly but does not guarantee optimal convergence to the RPCA problem.

The matrix completion problem can be seen as a special form of the matrix recovery problem, where we need to recover the missing values of the matrix given limited number of known entries. Similar to the logic in program (2), we then reformulate the program (8) as the RPCA matrix completion program defined as:

$$\text{minimize} \quad \|L\|_* \quad \text{subject to} \quad L + S = M, \quad \pi_\Omega(S) = 0 \tag{25}$$

where $\pi_\Omega : \mathbb{R}^{m \times n} \to \mathbb{R}^{m \times n}$ is a linear operator keeping the entries in Ω unchanged and setting those outside Ω as zeros [8]. Since S will compensate for the unknown entries of M are set as zeros, then the partial augmented Lagrangian function of (25) is:

$$L(L, S, Y, \mu) = \|L\|_* + \langle Y, M - L - S \rangle + \frac{\mu}{2} \|M - L - S\|_F^2 \tag{26}$$

Then similarly we can have the exact and inexact ALM methods for the matrix completion problem, where for updating S the constraint $\pi_\Omega(S) = 0$ should be imposed when minimizing

$L(L, S, Y, \mu)$. And finally we introduce the inexact ALM approach for the matrix completion problem in Algorithm 1 [8].

Input: Observation samples D_{ij}, $(i,j) \in \Omega$, of matrix $D \in \mathbb{R}^{m \times n}$.
1: $Y_0 = 0$; $E_0 = 0$; $\mu_0 > 0$; $\rho > 1$; $k = 0$.
2: **while** not converged **do**
3: // Lines 4-5 solve $A_{k+1} = \arg\min_A L(A, E_k, Y_k, \mu_k)$.
4: $(U, S, V) = \text{svd}(D - E_k + \mu_k^{-1} Y_k)$;
5: $A_{k+1} = U \mathcal{S}_{\mu_k^{-1}}[S] V^T$.
6: // Line 7 solves $E_{k+1} = \arg\min_{\pi_\Omega(E)=0} L(A_{k+1}, E, Y_k, \mu_k)$.
7: $E_{k+1} = \pi_{\bar{\Omega}}(D - A_{k+1} + \mu_k^{-1} Y_k)$.
8: $Y_{k+1} = Y_k + \mu_k(D - A_{k+1} - E_{k+1})$; $\mu_{k+1} = \rho \mu_k$.
9: $k \quad k+1$.
10: **end while**
Output: (A_k, E_k).

Figure 1: **Algorithm 1 (Matrix Completion via the Inexact ALM Method)**

2.3 Dual Method [8]

The dual method approaches the program (8) of RPCA problem via its dual. One first solves the dual problem:

$$\max_Y \langle M, Y \rangle, \quad \text{subject to} \quad J(Y) \leq 1, \tag{27}$$

for the optimal Lagrange multiplier Y, where

$$\langle A, B \rangle = \text{tr}(A^T B), \quad J(Y) = \max(\|Y\|_2, \lambda^{-1} \|Y\|_\infty), \tag{28}$$

and $\|\cdot\|_\infty$ is the maximum absolute value of the matrix entries. We could apply the steepest ascend algorithm constrained on the surface $\{Y | J(Y) = 1\}$ to deal with (27), where the constrained steepest ascend direction is obtained by projecting M onto the tangent cone of the convex body $\{Y | J(Y) \leq 1\}$. The advantage of the dual method is that the principle singular space associated to the largest singular value 1 is needed, and theoretically speaking, computing the special principal singular space should be easier than computing the principal singular space associated to the unknown leading singular values [8]. In general, the dual method is promising if we could obtain an efficient method for computing the principal singular space associated to the known largest singular space. The algorithm for RPCA problems via dual method is shown in Algorithm 2 [9], but we will not discuss derivation of algorithm in detail.

3 Data Implementation and Performance Comparison

The movie rating dataset contains over 100 million ratings from 480 thousand randomly-chosen, anonymous Netflix customers over 17000 movies. The ratings are on a scale from 1 to 5 (integer) to reflect their personal preference for a specific movie. To boost simplicity, here we extract first 25 percent of the data to mimic the movie recommendation system using the three algorithms mentioned in the above sections: singular value thresholding, augmented Lagrange multiplier, and dual method. However, due to the limited RAM space and high

Input: Observation matrix $D \in \mathbb{R}^{m \times n}$, λ.

$Y_0 = \text{sgn}(D)/J(\text{sgn}(D))$; $k \leftarrow 0$.
while not converged **do**
  Compute the projection D_k of D onto $N(Y_k)$:
  if $\|Y_k\|_2 > \lambda^{-1}|Y_k|_\infty$ **then**
    $D_k \leftarrow \pi_2(D)$, $A \leftarrow D$, $E \leftarrow 0$.
  else if $\lambda^{-1}|Y_k|_\infty > \|Y_k\|_2$ **then**
    $D_k \leftarrow \pi_\infty(D)$, $A \leftarrow 0$, $E \leftarrow D$.
  else
    $A \leftarrow 0$, $E \leftarrow 0$.
    while not converged **do**
      $A \leftarrow \pi_2(D - E)$, $E \leftarrow \pi_\infty(D - A)$.
    end while
    $D_k \leftarrow A + E$.
  end if
  Do line search to determine a step size δ_k.
  $Y_{k+1} \leftarrow \frac{Y_k + \delta_k(D - D_k)}{J(Y_k + \delta_k(D - D_k))}$ and $k \leftarrow k + 1$.
end while

Output: (A, E).

Figure 2: **Algorithm 2 (Robust PCA via the Dual)**

sparsity of the data matrix, we cannot conduct the experiment using the above algorithms successfully, especially for the dual method, which is a relatively slow and complex algorithm. As a result, we turn to the data of real-life settings, which is a dataset I designed on my own, collecting rating information on 23 movies from over 20 friends from all walks of life. The advantages of the dataset compared to Netflix dataset include: low sparsity and small shape which are easy for implementation and training, and also the movies are carefully categorized into three type: Romance, Disaster, Science Fiction. So we could also see the relationship between different categories. From the data excerpt 4, we could see the ratings ranging from 1 to 5 (integer), and 0 means that he or she did not watch the movie.

Then, we apply the three algorithms to the data matrix and acquire the completed low-rank matrix of the same size as before. Theoretically speaking, Inexact ALM is the fastest algorithm and the most accurate one while SVT is the slowest and least accurate one in the default setting 3. For each row (customer), we sort the trained rating results of movies never watched before from highest to the lowest (most likely preferred to least likely preferred) and get the scatter plot of three rankings of movies using three algorithms for the first 6 customers 3.

The y-axis is the movie id. The highest horizontal line is the division between science fiction (18-22) and disaster (11-17), and the lowest line is the division between disaster and romance (0-10). The x-axis is the rank number. From the plots, we could see that the results generated from the three algorithms are quite different: some movies' rankings from three algorithms are similar or even overlap while some are separated far between each other. These might be due to the small size of dataset and the low maximum iteration times limited by the computer condition, and also there might be some slightly different constraints for different algorithms (the tolerance of error for dual method is distinctive with others') because the codes for three algorithms are not unified. With larger size of data matrix and better computer condition, we might get more statistically reliable and explicable results. However, one thing worth noting is that even rankings for some specific movies are quite different using three

Robust PCA Algorithm Comparison

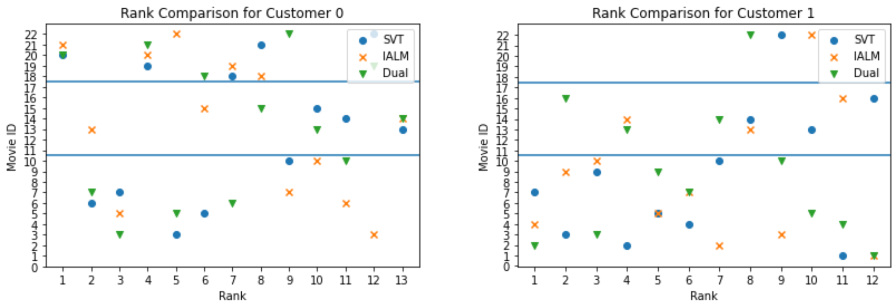
Algorithm	Rank of estimate	Relative error in estimate of A	Time (s)
Singular Value Thresholding	20	3.4×10^{-4}	877
Accelerated Proximal Gradient	20	2.0×10^{-5}	43
Accelerated Proximal Gradient (with partial SVDs)	20	1.8×10^{-5}	8
Dual Method	20	1.6×10^{-5}	177
Exact ALM	20	7.6×10^{-8}	4
Inexact ALM	20	4.3×10^{-8}	2
Alternating Direction Methods	20	2.2×10^{-5}	5

Figure 3: Robust PCA Algorithm Comparison [10]. Note that the table represents typical performance, using default parameters, on random matrices drawn according to the distribution specified earlier. The performance could vary when dealing with matrices drawn from other distributions or with real data.

num	lmjr	ccnn	ldz	lsjr	tmm	cqsl	amypj	hdlq	bwbj	rsnzdxx	lsn	ht	fsx	hxqj	mrwt	dlr	tsddz	ylzj	ydm	hxldnyy	dmkj	hkdg	yt
1	4	2	5	0	4	0	0	0	5	3	0	3	4	0	0	0	4	3	0	0	0	0	0
2	4	0	0	0	0	0	5	0	5	0	0	4	5	0	0	3	0	3	4	4	5	4	0
3	4	3	5	5	5	5	5	5	5	5	5	4	5	0	4	4	4	0	4	4	5	0	0
4	5	5	3	0	0	0	4	0	0	0	0	5	4	4	0	5	4	0	4	0	5	5	0
5	4	2	4	4	4	0	0	3	5	0	3	0	4	2	0	0	5	3	3	0	4	4	4
6	5	0	0	5	4	0	0	5	4	0	0	0	0	0	0	0	3	4	0	0	0	0	0

Figure 4: Original Ranking before completion

algorithms, many movies of the same ranking may be in a cluster: in the same movie type. This might be a signal that although the movie recommending system is not as accurate as we imagine, it could still recommend us movies of similar topic or same category. Improvement will be carried out later.



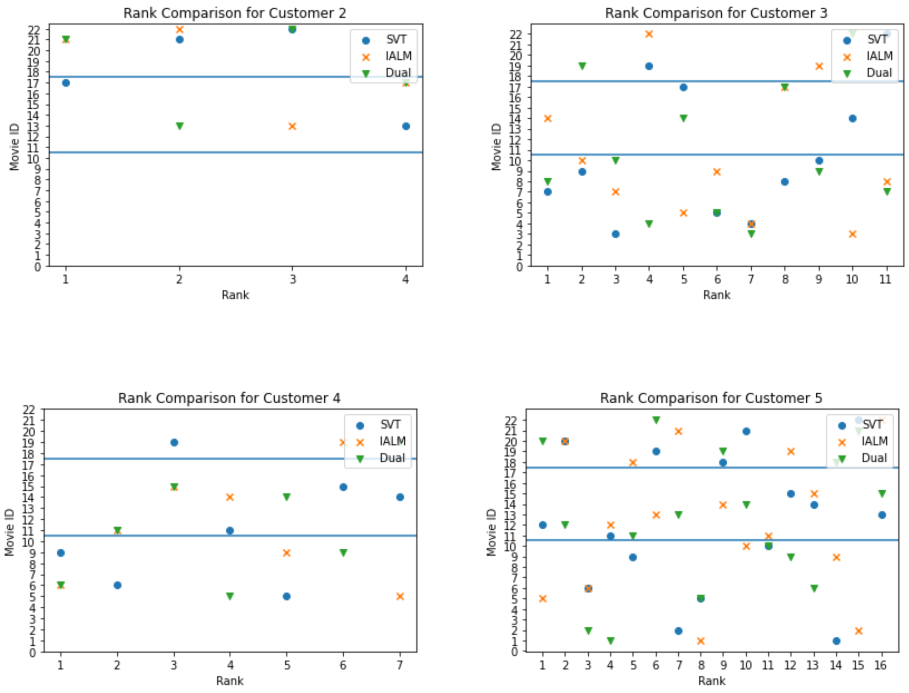


Figure 5: Algorithm Performance Comparison for 6 Customers. And the three minimized nuclear norms are 50, 49 and 68 respectively, which are relatively close to show that the algorithms work very well.

4 Sparse PCA Analysis

Skipping the analysis for the special square matrix case, we apply the same reasoning to the case of a non-square matrix $A \in \mathbb{R}^{m\times n}$, and the sparse PCA program can be written as [4]:

$$\begin{array}{ll} \text{maximize} & u^T A v \\ \text{subject to} & \|u\|_2 = \|v\|_2 = 1 \\ & \mathbf{Card}(u) \le k_1, \quad \mathbf{Card}(v) \le k_2 \end{array}$$

and the variables $(u, v) \in \mathbb{R}^m \times \mathbb{R}^n$ where $k_1 \le m$, $k_2 \le n$ are fixed integers to control the sparsity. The semidefinite relaxation is:

$$\begin{array}{ll} \text{maximize} & \mathbf{Tr}(A^T X^{12}) \\ \text{subject to} & X \succeq 0, \ \mathbf{Tr}(X^{ii}) = 1 \\ & \mathbf{1}^T |X^{ii}| \mathbf{1} \le k_i, i = 1, 2 \\ & \mathbf{1}^T |X^{12}| \mathbf{1} \le \sqrt{k_1 k_2} \end{array}$$

in the variable $X \in \mathbf{S}^{m+n}$ with blocks X^{ij} for i, j=1, 2. But how to deal with the sparse decomposition? The paper [4] mentions the sparse decomposition for a matrix $A \in \mathbf{S}^n$, a given symmetric matrix (square case). Our intention is to decompose it in factors with targeted

sparsity k. Then we solve the semidefinite relaxed problem:

$$\begin{aligned} \text{maximize} \quad & \mathbf{Tr}(A_1 X) \\ \text{subject to} \quad & X \succeq 0 \\ & \mathbf{Tr}(X) = 1 \\ & \mathbf{1}^T \|X\| \mathbf{1} \leq k \end{aligned}$$

Let X_1 denote the solution, we truncate X_1, retaining only the dominate eigenvector x_1. Eventually, we deflate A_1 to acquire:

$$A_2 = A_1 - (x_1^T A_1 x_1) x_1 x_1^T, \tag{29}$$

and we iterate the process to obtain further components. In the general PCA problem, the decomposition stops when $\mathbf{Rank(A)}$ factors have been found because $A_{\mathbf{Rank(A)}+1}$ is then equal to zero. However, in the scenario of the sparse PCA decomposition, we have no guarantee that this condition would occur. Consequently, we can add an additional set of linear constraints $x_i^T X x_i = 0$ to problem 4 to enforce the orthogonality of $x_1, \cdots, x_n$ and the decomposition will stop after a maximum of n iterations. Since our experiment in section 3 is a relative small problem, we then can apply the interior-point solvers such as SEDUMI [11] and SDPT3 [12] to solve the program efficiently, to find the principal components with sparsity.

5 Conclusion and Discussion

This paper provides a comprehensive analysis of matrix completion program from scratch, including the basic introduction, typical algorithms analysis and real-data experiment, and it is a simple hodgepodge for many famous papers in this field of study.

In conclusion, in the project, there are mainly two branches for matrix completion problem: nuclear norm minimization and robust principal component analysis, which are essential to the movie recommendation system. For nuclear norm minimization program, we apply singular value thresholding algorithm to solve the optimization problem, calculating the low-rank completed matrix of the original size. For the robust principal component analysis, we implement both inexact augmented Langrange multiplier method and dual method to the former sparse data matrix, both of which prove to be a conducive algorithm to deal with matrix completion and recovery problem. Though the outcomes of the real-data implementation is not as successful as we imagine: the Netflix dataset is too sparse and large to implement and the results of my self-designed dataset is not statistically well explained, they still demonstrate the possibility and effectiveness of these algorithms in some simplified and specified settings. With all this well-founded and sound theoretical analysis, including sparse principal components analysis, of matrix completion and the movie recommendation system in hand, we believe some improvement will be realized in the near future.

References

[1] Cai, J.-F., Candes, E. J., and Shen, Z. A singular value thresholding algorithm for matrix completion.

[2] Candès, E. J., Li, X., Ma, Y., and Wright, J. Robust principal component analysis? *CoRR abs/0912.3599* (2009).

[3] Candes, E. J., and Recht, B. Exact matrix completion via convex optimization.

[4] d'Aspremont, A., Ghaoui, L. E., Jordan, M. I., and Lanckriet, G. R. G. A direct formulation for sparse PCA using semidefinite programming. *CoRR cs.CE/0406021* (2004).

[5] Jolliffe, I. Principal component analysis: A beginner's guide - i. introduction and application. *Weather 45* (10 1990), 375–382.

[6] Jolliffe, I. T., Trendafilov, N. T., and Uddin, M. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics 12*, 3 (2003), 531–547.

[7] Journée, M., Nesterov, Y., Richtárik, P., and Sepulchre, R. Generalized power method for sparse principal component analysis.

[8] Kuybeda, O., Frank, G. A., Bartesaghi, A., Borgnia, M., Subramaniam, S., and Sapiro, G. A collaborative framework for 3d alignment and classification of heterogeneous subvolumes in cryo-electron tomography. *Journal of Structural Biology 181*, 2 (feb 2013), 116–127.

[9] Lin, Z., Ganesh, A., Wright, J., Wu, L., Chen, M., and Ma, Y. Fast convex optimization algorithms for exact recovery of a corrupted low-rank matrix.

[10] Ma, Y. Sample code for robust pca.

[11] Sturm, J. F. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software 11*, 1-4 (1999), 625–653.

[12] Toh, K. C., Todd, M., and Tütüncü, R. H. Sdpt3 – a matlab software package for semidefinite programming. *OPTIMIZATION METHODS AND SOFTWARE 11* (1999), 545–581.