Robust PCA Matrix Completion in Non-Uniform Observations

Vegen Soopramanien
vs385

Howward Shih
hs985

May 2019

1 Introduction

Principal Component Analysis is a widely recognized operation in computational data analysis with application ranging from web search to computer vision or image analysis. The goal of PCA is to find a low-rank subspace that best approximates a data matrix $\mathbf{X} \in R^{n \times d}$. Standard method of doing PCA involves computing the Singular Value Decomposition, which unfortunately fails in modern data applications problems owing to missing and corrupted entries in the input matrix; not to mention that SVD is highly sensitive to outliers. This follows from the idealized assumption of small noise that is i.i.d. Gaussian.

In line with these problems, Robust PCA (RPCA) and it's applications to modern data analysis have become much more relevant, with a vast amount of contribution in the recent years [1], [3]. Consider the following setting for RPCA: Given a data matrix $\mathbf{X} \in R^{nxd}$ that has decomposition $\mathbf{X} = \mathbf{L_0} + \mathbf{S_0}$ where L is a rank r matrix and S is a sparse (possibly corrupted) matrix whose nonzero entries have arbitrarily large magnitude. The brittleness of PCA to corrupted entries jeopardizes it's motivation: in particular, a single major corrupted entry in X could result in an estimate of the low-rank subspace $\hat{\mathbf{L}}$ to be arbitrarily different from the true $\mathbf{L_0}$. The aim of RPCA is therefore to recover a low-rank matrix $\mathbf{L_0}$ from highly corrupted observations $\mathbf{X} = \mathbf{L_0} + \mathbf{S_0}$; while classical PCA assumes small Gaussian noise, under RPCA, the entries of $\mathbf{S_0}$ can have arbitrarily large magnitudes (modeling grossly corrupted observations is indeed common in visual and bioinformatic data), with sparse (hence affecting only a fraction of the entries of X) and unknown support. This, in fact, makes the problem even more complicated than the matrix completion problem that has been studied over the recent years.

The problem of Matrix Completion (MC) involves dealing with a situation when some number of columns (or possibly rows) of a given matrix are completely and arbitrarily corrupted. For a direct application, consider Collaborative Filtering methods where given partial observation of users' preferences, the

goal is to accurately recover predictions for the users' unrevealed preferences. However, the problem is that sometimes, malicious users try to manipulate the predictions of the algorithm by altering their inputs. And when even just a few columns of the matrix are corrupted, the output of CF algorithms tend to differ significantly from the true matrix. The goal of MC is to successfully render predictions for 'honest' users given a matrix of corrupted and partially observed data, while simultaneously identifying the malicious users.

Surprisingly, there exists a strong connection between solving the Robust PCA problem (i.e. finding the best rank-k subspace given a matrix X of grossly corrupted observations), and dealing with the Matrix Completion problem (i.e. completing unobserved entries, and identifying corrupted columns), and we will present this in the following section.

If we think of the corruption of certain columns of a matrix as the addition of a column-sparse matrix to a low-rank matrix, then the MC problem in a way is similar to the RPCA formulation of Candès et al. [1] Specifically, given only partial observation of the matrix M, the objective is to recover

$$M = L_0 + C_0$$

where L_0 is low-rank and C_0 only has a few nonzero columns. Chen et al [2] propose a robustification of the MC problem in investigating Robust Collaborative Filtering and RPCA with partially observed data. They propose the following setup: given a $p \times n$ matrix M, among the n columns, a fraction $1-\gamma$ of them span an r-dimensional subspace of $\mathbf{R}^p$ and the remaining γn columns are arbitrarily corrupted. Then, given only partial observation of the matrix M, the objective is to infer the true subspace of the non-corrupted columns while at the same time recovering the identities of the corrupted ones. Thus, using the above decomposition, L_0 is rank-r, and at most $(1-\gamma)n$ of the columns of L_0 are non-zero. At most γn of the columns of C_0 are non-zero. Let $\Omega \subseteq [p] \times [n]$ be the set of indices of observed entries of M and P_Ω the orthogonal projection onto the linear subspace of matrices supported on Ω, such that

$$P_\Omega(X) = X_{ij} \text{if } (i,j) \in \Omega, \text{ and } P_\Omega(X) = 0 \text{ otherwise}$$

The goal then is to recover the exact subspace of L_0 and the locations of the nonzero columns of C_0, given $P_\Omega(M)$. Chen et al. [2] achieve this by solving a tractable convex program using their *Manipulator Pursuit* algorithm. In particular, given as inputs $P_\Omega(M), \Omega, \lambda$, we solve for the optimum (L^*, C^*) in:

$$\arg\min_{L,C} \|L\|_* + \lambda \|C\|_{1,2} \text{ subject to } P_\Omega(L+C) = P_\Omega(M)$$

The *Manipulator Pursuit* (MP) algorithm is clearly very similar to the *PCP* algorithm proposed by Candès et al [1], and the goal of our research topic leverages on that strong relationship. We note however that the distinction is that while *PCP* assumes the support of the sparse matrix is uniformly random, *MP* doesn't and instead specifically deals with corrupted columns.

2 Background

2.1 Robust Principal Component Analysis

Recent work by Candès et Al [1] defined RPCA as a pure matrix decomposition problem which they named *Principal Component pursuit* (PCP): given a matrix $\mathbf{X} \in R^{nxd}$, the objective is to recover a low-rank matrix $\mathbf{L_0}$ whose columns subspace gives the principal components, and a sparse matrix $\mathbf{C_0}$ of outliers; in [1], Candès et Al. proved that the PCP estimate that solves the problem:

$$\arg\min_{L,C} \|L\|_* + \lambda \|C\|_1 \quad s.t. \;\; L + C = \mathbf{X}$$

will exactly recover the low-rank $\mathbf{L_0}$ and the sparse $\mathbf{C_0}$. Note here that $\|M\|_* := \sum_i \sigma_i(M)$ denotes the nuclear norm of a matrix M, and $\|M\|_1 := \sum_{ij} |M_{ij}|$ denotes the ℓ_1-norm of M.

An important assumption of PCP is the identifiability between the $\mathbf{L}$ and $\mathbf{C}$ component: in particular, we assume that $\mathbf{L}$ is low-rank and while its entries can be dense, they cannot be sparse. Similarly, we assume that $\mathbf{C}$ cannot be low-rank. One way of enforcing this assumption as suggested by Candès et Al. is to introduce a notion of incoherence on the low-rank component (we require $\mathbf{L}$'s left and right singular vectors to be "dense" or "incoherent" with respect to a sparse vector). We give further details on incoherence below.

In this study, we will focus on sparse matrix $\mathbf{C}$ with particular structures - sparse matrix as column corruptions. This setting is important in defining the problem. Sparse entries are "corrupted" only if they disrupts the algorithm's ability to recover the non-corrupted entries. Therefore, if we introduce $\mathbf{C}$ as sparse entries uniformly distributed with large magnitude, then the corruption may or may not affect the recovery algorithm. On the other hand, if entire columns are corrupted, there is a much higher chance of the algorithm successfully identifying the corruptions. Therefore to narrow the scope of the investigation, we will only focus on sparse matrix as corrupted columns instead of entries. However, this prompts future investigation on the effect of corrupted entries on matrix completion and decomposition algorithms.

Corrupted columns alter the objective function of RPCA slightly. Whereas in corrupted entries we are interested in the ℓ_1 norm $\|.\|_1$, in corrupted columns we are interested in ℓ_1 norm of the ℓ_2 norms of the columns, i.e. $\|.\|_{1,2}$.

2.2 Augmented Lagrange Multiplier

In Candès et Al [1], the algorithm proposed to solving the convex *Principal Component Pursuit* problem was the method of Augmented Lagrange Multipliers (ALM). With higher proven accuracy and no required tuning of parameters, ALM also has

better stability. It operates on the *augmented Lagrangian*:

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

Let $C_\tau : \mathbf{R} \to \mathbf{R}$ denote the shrinkage operator $C_\tau[x] = sgn(x)max(|x|-\tau, 0)$ and we extend it to matrices by applying it to each element. Candès et al then show that:

$$\arg\min_S l(L, C, Y) = S_{\lambda\mu^{-1}}(M - L + \mu^{-1}Y)$$

Given a matrix X, let $D_\tau(X)$ denote the singular value thresholding operator defined by $D_\tau(X) = US_\tau(\Sigma)V^*$ where $X = U\Sigma V^*$ is any SVD of X. Candès et al then prove it is easy to show that:

$$\arg\min_L l(L, C, Y) = D_{\mu^{-1}}(M - C + \mu^{-1})Y$$

This leads them to develop this very practical strategy therefore of first minimizing l with respect to L (fixing C), then minimize l with respect to C (fixing L), and then finally updating the Lagrange multiplier matrix Y based on the residual $M - L - C$.

We implemented the ALM algorithm as in Chen et Al [2], which introduces an additional step in ALM to accommodate for missing entries, extending RPCA to a robust matrix completion problem. Applying the algorithm to a 40x40 matrix with 3 corrupted columns, we were able to recover the low rank matrix and corrupted columns. However, we find that ALM is very sensitive to hyper-parameter and requires very fine tuning to separate the matrices. If λ is too small, then sparse component fails to converge and grows infinitely large. If λ is too big, then the thresholding function would fail to retain any corrupted columns. An observation in the numerical experiment is that C is hard to recover, and when the columns are recovered, the values of the entries are smaller by orders of magnitude. However, the algorithm is able to recover the low rank component L with lower sensitivity to λ. This indicates that while the algorithm cannot successfully recover the exact values of the L matrix, it is able to recover the missing entries after removing the corruption.

This discrepancy actually makes intuitive sense. Consider I_c as in index of the corrupted columns, and I_c^c as the index of the authentic columns, and I as the whole index. Now consider two corruption schemes - the first is to generate a full authentic matrix on all I columns, and then introduce the corrupted columns by superimposing them; the second is to only generate authentic matrix on columns in L_c^c and leave the rest as zeros. The algorithm has no way of distinguishing the two schemes. In either case, the algorithm will try to recover the column as an authentic column that lies in the vector space of the other authentic column plus a sparse noise. Therefore, it is more important to be able to identify the corrupted columns rather than precisely recover them.

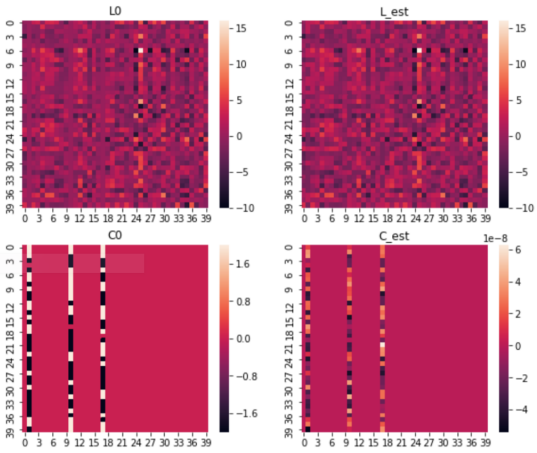

Figure 1: ALM recovery. $L0$ and $C0$ represents original matrices, and L_{est} and C_{est} represents the recovered components.

2.3 Nuclear Norm Matrix Completion

Given a matrix Y, and a set of observations Ω, the objective of nuclear norm minimization matrix completion is to minimize either of the following objective functions.

$$\|X\|_* \quad s.t. \quad X_\Omega = Y_\Omega$$

$$\|X_\Omega - Y_\Omega\|_F^2 + \lambda \|X\|_*$$

Because the underlying assumption is that Y has low rank, naturally minimizing the rank given the constraint or penalizing the rank should recover the Y. However, minimizing the rank is not a convex optimization problem, minimizing the nuclear norm is a convex surrogate. Candès et Al [1] has shown that this objective function can successfully recover the matrix. To give a more intuition, consider the rank of the matrix as a function of its singular values σ_i.

$$Rank(Y) = \sum I(\sigma_i > 0)$$

$$\|Y\|_* = \sum \sigma_i(Y)$$

If we consider Y as the product of two lower dimensional matrix U and V, i.e. $Y = U^T V$, $U \in R^{kxm}$, $V \in R^{kxn}$, then the nuclear norm of Y can be converted to the trace norm of Y [4].

$$\|Y\|_* = \|Y\|_{tr} = \arg\min_{Y=U'V} \frac{1}{2}(\|U\|_F^2 + \|V\|_F^2)$$

Therefore, the objective function to be optimized becomes

$$\mathcal{L} = \sum_{i,j}(Y_{i,j} - U_i V_j^T)^2 + \frac{\lambda}{2}\Big(\|U_i\|^2 + \|V_j\|^2 \Big)$$

Gradient descent with the correct λ parameter is able to recover the full matrix:

$$\frac{\partial \mathcal{L}}{\partial U_i} = \sum_{j|i,j\in\Omega} -2(Y_{i,j} - U_i V_j^T)V_j + \lambda \sum_{j|i,j\in\Omega} U_i$$

$$\frac{\partial \mathcal{L}}{\partial V_i} = \sum_{i|i,j\in\Omega} -2(Y_{i,j} - U_i V_j^T)U_j + \lambda \sum_{i|i,j\in\Omega} V_j$$

2.4 Matrix Coherence

On of the sufficient conditions for matrix recovery using nuclear norm minimization is that the matrix is incoherent. Incoherent matrices are those whose "information" are uniformly distributed across the entries. Conversely, a coherent matrix is one who has "important information" concentrated in a few entries. The formal definition of local coherence of a matrix is the following. Consider the SVD of Y, $Y = U\Sigma V^T$

$$\mu_i \quad s.t. \quad \|U_i\| = \sqrt{\frac{\mu_i r}{m}}$$

$$\nu_j \quad s.t \quad \|V_j\| = \sqrt{\frac{\nu_j r}{n}}$$

Where μ_i are the local coherence of the rows of Y, and ν_j are the local coherence of the columns of Y. Therefore, the overall coherence of the matrix is the maximum of μ_i, ν_j. Interestingly, the expression of local coherence is directly related to the statistical leverage of the rows and columns of a matrix, $\|U_i\|, \|V\|_j$. This is very similar to the components in nuclear norm minimization, and the connection to statistical leverage gives us some intuition as to why the incoherence assumption is required for successful matrix recovery.

In linear regression, a point that has high statistical leverage corresponds to high influence on the coefficient of the regression line. If we consider the rows or matrix as observed data points, then observing points with higher leverage score can contribute significantly to interpolating missing observations. On one hand, this means that given a coherent matrix with uniform entry observation probability, if we do not observe points with high local coherence, the missing entries may not be successfully recovered. On the other hand, if we observe entries that have high local coherence, we may need fewer observations to recover the full matrix.

The following figure is a numerical demonstration on coherence effect of matrix completion at different observation level. As we can see, high coherence hinders matrix recovery even if a significant percentage of entries are observed.

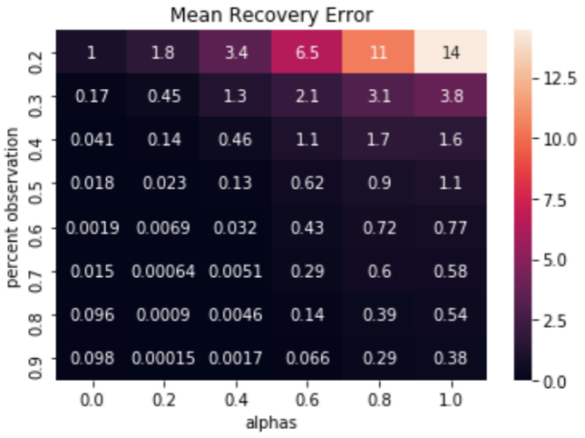

Figure 2: Mean recovery error of matrices at different observation and coherence level

2.5 Non-uniform Observation

Another sufficient condition for successful nuclear norm matrix completion is that the random observations came from a sample of uniform distribution. One of the simplest explanation of why this assumption is necessary is that if one row or one column has no observation at all, then there is no way of recovering that row or column. Therefore, we expect each row or column to at least have some observation. To examine this further, we consider the the objective function of nuclear norm as regularization.

$$\mathcal{L} = \sum_{i,j \in \Omega} (Y_{i,j} - U_i V_j^T)^2 + \frac{\lambda}{2} \left(\|U_i\|^2 + \|V_j\|^2 \right)$$

$$= \sum_{i,j \in \Omega} (Y_{i,j} - U_i V_j^T)^2 + \frac{\lambda}{2} \sum_{i,j \in \Omega} \|U_i\|^2 + \frac{\lambda}{2} \sum_{i,j \in \Omega} \|V_j\|^2$$

$$= \sum_{i,j \in \Omega} (Y_{i,j} - U_i V_j^T)^2 + \frac{\lambda}{2} \sum_{i | i,j \in \Omega} n_i \|U_i\|^2 + \frac{\lambda}{2} \sum_{j | i,j \in \Omega} n_j \|V_j\|^2$$

Where $n_i = \sum_j I(i,j \in \Omega)$, $n_j = \sum_i I(i,j \in \Omega)$. Therefore, while the regularization constant λ is set for the entire matrix, each row and column gets penalized differently depending on how many times it gets observed. If the observations come from a uniform sample, then $n_i = c_1 \ \forall i$, $n_j = c_2 \ \forall j$. However, if the observations are not randomly sampled, then columns/rows with many observations will get penalized more than columns/rows with fewer observations. This may seem counter intuitive in practical application, such as collaborative filtering, since we should take advantage of users that have given many rating. On the other hand, since the purpose of regularization is

to mitigate over-fitting, we want to also prevent penalizing columns that are more frequently the same as penalizing those that are observed less frequently. Therefore, it is important to balance the over-penalization problem and the over-fitting problem simultaneously. This inspires the weighted nuclear norm minimization.

2.6 Weighted Nuclear Norm minimization

Relating local coherence structure to the nuclear norm minimization, we can rewrite the SVD decomposition of Y in the following way.

$$Y = U\Sigma V^T = U\sqrt{\Sigma}\sqrt{\Sigma}V^T = \tilde{U}\tilde{V}^T$$

Therefore, minimizing the trace norm, i.e.

$$\frac{1}{2}\left(\left\|\tilde{U}\right\|_F^2 + \left\|\tilde{V}\right\|_F^2 \right) = \frac{1}{2}\sum_{i,j} \sigma_i(Y) \left\|U_i\right\|^2 + \sigma_j(Y) \left\|V_j\right\|^2$$

Hence, the leverage of the i^{th}, j^{th} column/row are weighted by the i^{th}, j^{th} singular values, respectively. Conversely, in the nuclear norm minimization problem, the leverage becomes the weights for the singular values. For a square matrix, the leverage scores are just 1, but for a non rectangular matrix, the singular values are weighted by their corresponding leverage scores. Similar to the frequency weight for non-uniform observations, non-uniform local coherence/leverage can result in unbalanced penalization of the rows and columns. Rows/Columns with higher leverage scores are more severely penalized, hence the algorithm may not be able to recover the missing entries successfully. This further provides basis of why the nuclear norm minimization should be weighted. The weighted minimization objective function becomes the following.

$$\mathcal{L}_w = \sum_{i,j\in\Omega} (Y_{i,j} - U_i V_j^T)^2 + \frac{\lambda}{2}\left(p(i) \left\|U_i\right\|^2 + q(j) \left\|V_j\right\|^2 \right)$$

This expression allows us to penalize different components differently by weighing their importance and frequency of appearance. The weights $p(i)$, $q(j)$ are chosen such that the local coherence structures are aligned with the observation probability. In another expression, we transform the original matrix Y by RYC, where R and C are diagonal weight matrices, and rewrite the objective function.

$$\hat{X} = RXC, \quad \hat{Y} = RYC$$

$$min \left\|\hat{X}\right\|_* \quad s.t. \;\; \hat{X}_{i,j} = \hat{Y}_{i,j}, \quad \forall i,j \in \Omega$$

A simple way to choose $p(i)$, $q(j)$ is by some values proportional to their empirical estimates:

$$\hat{p_i} = \frac{\sum_j I(i,j \in \Omega)}{\sum_{k,j} I(k,j \in \Omega)} = \frac{n_i}{|\Omega|} \qquad \hat{p_j} = \frac{\sum_i I(i,j \in \Omega)}{\sum_{i,k} I(i,k \in \Omega)} = \frac{n_j}{|\Omega|}$$

Intuitively, we want to down weight the penalty on rows and column with more observations, hence we choose them to be inversely proportional to the empirical estimates

$$\mathcal{L}_{w_{partial}} = \sum_{i,j \in \Omega} (Y_{i,j} - U_i V_j^T)^2 + \frac{\lambda}{2|\Omega|} \left(n_i^{\alpha-1} \|U_i\|^2 + n_j^{\alpha-1} \|V_j\|^2 \right)$$

$$= \sum_{i,j \in \Omega} (Y_{i,j} - U_i V_j^T)^2 + \frac{\lambda}{2|\Omega|} \sum_{j|i,j \in \Omega} n_i^{\alpha} \|U_i\|^2 + \sum_{j|i,j \in \Omega} n_j^{\alpha} \|V_j\|^2$$

In this expression the counts for rows and column appearance are first normalized by the total number of observations $|\Omega|$. $\alpha \in [0,1]$ is the adjustment constant, where $\alpha = 0$ corresponds to unweighted, and $\alpha = 1$ corresponds to un-adjusted weighted minimization. Unweighted minimization means that every row is equally penalized, which might results in over-fitting, whereas un-adjusted weighted minimization can result in over-penalization.

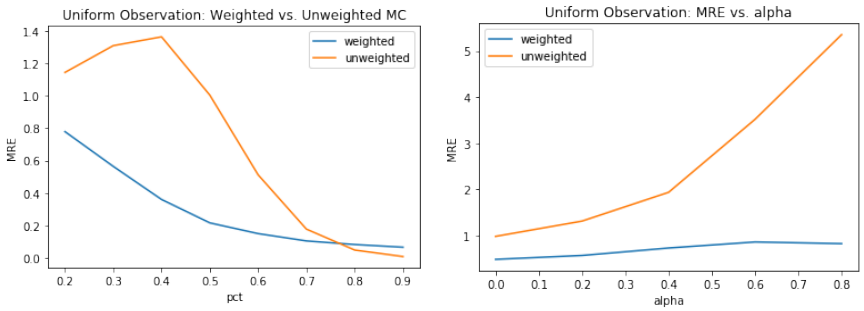

Figure 3: Weighted vs. Unweighted MC: the figure on the right shows weighted matrix completion's superior performance over the unweighted in different observation percentage. The figure on the right shows similar results over coherence.

Figure 3 shows that weighted($\alpha = 1$) nuclear norm minimization performs better than unweighted($\alpha = 0$) even in uniform observation. This trend is consistent across observation percentage and coherence of the matrices. Although theoretically they should perform the same, since uniform observation implies that the weights are also uniform, but from the previous derivation of the objective function, this discrepancy in performance is likely due to the penalty weight λ. However, in Figure 4 we see that weighted optimization again performs consistently better than unweighted across the board when the observations are non-uniform. This reinforces that weighted optimization outperforms the benchmark in every setting, which makes sense theoretically.

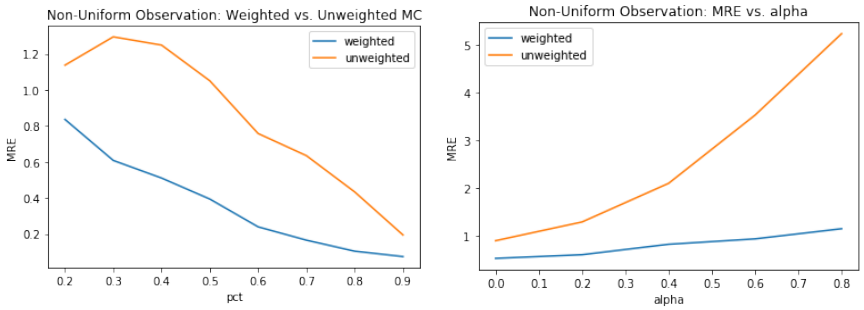

Figure 4: Weighted vs. Unweighted MC: Non-uniform Observations

3 Main Results

3.1 Integrating RPCA and Weighted NNM

The objective functions of RPCA and nuclear norm matrix completion are very similar. In fact, without the weighted nuclear norm RPCA's objective function is just an extension to that of the nuclear norm matrix completion with an additional sparse component penalty.

$$Y = L + S = U^T V + S$$

$$Y_\Omega = L_\Omega + S_\Omega = (U^T V)_\Omega + S_\Omega$$

$$\pounds_{rpca+wnnm} = \sum_{i,j \in \Omega} (Y_{i,j} - U_i^T V_j - S_{ij})^2 + \frac{\lambda_1}{2} \left(p(i) \left\| U_i \right\|^2 + p(j) \left\| V_j \right\|^2 \right) + \lambda_2 \left\| S_{.j} \right\|_2$$

With the additional sparse component, the objective function becomes more tricky to optimize. With only the matrix completion problem, weights can be added in the gradient descent, but the sparse component is difficult to recover using only gradient descent. For only RPCA, ALM is able to recover sparse component by thresholding entries of column norm, but introducing the weights is less straightforward. However, we can easily adapt the thresholding technique in our gradient descent algorithm. Mainly through the Algorithm 1. The $\Lambda_\gamma(.)$ function is the thresholding function that sets the $n\gamma$ columns with the smallest l2 norm to zero.

Introducing the partial weighted nuclear norm, we derive the following gradients

$$\frac{\partial \pounds}{\partial U_i} = \sum_{j|i,j \in \Omega} -2(Y_{i,j} - U_i^T V_j - S_{ij}) V_j + \lambda_1 p(i)^\alpha \sum_{j|i,j \in \Omega} U_i$$

Algorithm 1 Gradient Descent with Thresholding

1: Initialize U, V, S as random matrices of size kxm, kxm, mxn.
2: Update $U_{(k+1)} = U_{(k)} - \eta \Delta \mathcal{L}_{U_{(k)}}$
3: Update $V_{(k+1)} = V_{(k)} - \eta \Delta \mathcal{L}_{V_{(k)}}$
4: Update $S_{(k+1)} = \Lambda_\gamma \Big(S_{(k)} - \eta \Delta \mathcal{L}_{S_{(k)}} \Big)$
5: k = k+1
6: Repeat step 2 to 5 until convergence or max iteration.

$$\frac{\partial \mathcal{L}}{\partial V_i} = \sum_{i|i,j \in \Omega} -2(Y_{i,j} - U_i^T V_j - S_{ij}) U_j + \lambda_1 p(j)^\alpha \sum_{i|i,j \in \Omega} V_j$$

$$\frac{\partial \mathcal{L}}{\partial S_{ij}} = \sum_{i,j \in \Omega} -2(Y_{ij} - U_i^T V_j - S_{ij}) + \lambda_2 Sign(S_{ij})$$

This algorithm also easily allows stochastic gradient descent implementation for more efficient computation in larger matrices. We refer to this method as *robust weighted matrix completion.*

3.2 Numerical Experiments

3.2.1 Synthetic Datasets

We initially run and test our implementations on some synthetic dataset. We replicate the setting used by Chen et Al [5]. We generate an $m \times n$ matrix $\mathbf{X} = \mathbf{L}^* + \mathbf{S}^* \in R^{m \times n}$. The low-rank part is given by $\mathbf{L}^* = \mathbf{U}^T \mathbf{V}$, where $\mathbf{U}, \mathbf{V}$ have entries drawn independently from a standard Gaussian distribution. To generate the sparse component $\mathbf{S}$, we first uniformly sample γ columns to be the corrupted column. For a given sparsity parameter β, each entry of $\mathbf{S}^*$ is set to be nonzero with probability β, and the values of the nonzero entries are uniformly sampled from $[\frac{-5r}{d}, \frac{5r}{d}]$.

To generate a coherent matrix $\mathbf{M}$ with rank k, we repeat the same procedure with the dot product of two Gaussian matrices. Additionally, we add a power decay diagonal matrix $\mathbf{D}$ s.t. $D_{ii} = i^{-\alpha}$, where α is the decay constant. The closer α is to one, the more coherent it is.

Non-uniform sampling probabilities are generated from the Dirichlet distribution with parameter α_i. We set the α_i to be constant across i. The smaller α_i is, the more non-uniform the distribution.

3.2.2 Metric

Matrix recovery error is generally used as the metric for evaluating performance.

$$MRE = \frac{\left\| M - \hat{M} \right\|_F}{\|M\|_F}$$

For robust matrix recovery, as discussed earlier, the algorithm can only recover the column index of the corrupted columns, but not the values themselves. Moreover, as we are less interested in recovering the corrupted columns, measuring the precision of this matrix is not as important. Therefore, let I_C be the index of the corrupted columns, and $\hat{I_C}$ be the index for the recovered corrupted columns. Note $I_C = \{j \mid \|C_{.j}\| > 0\}$. We can then measure the recovery error for the corrupted columns as

$$MRE_C = \frac{|I_C \bigcup \hat{I_C}| - |I_C \bigcap \hat{I_C}|}{|I_C|}$$

For the low rank matrix, we will ignore the corrupted columns when calculating the MRE. Using the column index set I_C^c

$$MRE_L = \frac{\left\| L_{I_C^c} - \hat{L}_{I_C^c} \right\|_F}{\left\| L_{I_C^c} \right\|_F}$$

3.2.3 Matrix Completion with Corrupted Columns and non-uniform observation

In this experiment we investigate the performance of robust matrix completion, weighted matrix completion, and robust weighted matrix completion on matrices that have corrupted columns and non-uniform observation. The results are shown in figure 5.

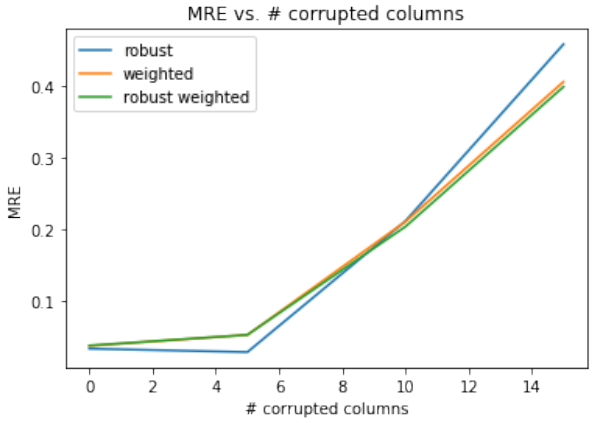

Figure 5: Mean recovery error of matrices at different corruption level in uniform observation

We see a surprising result in this experiment in that the three methods perform about the same. This indicates that the robust matrix completion is not correctly removing the corrupted entries despite being able to identify them.

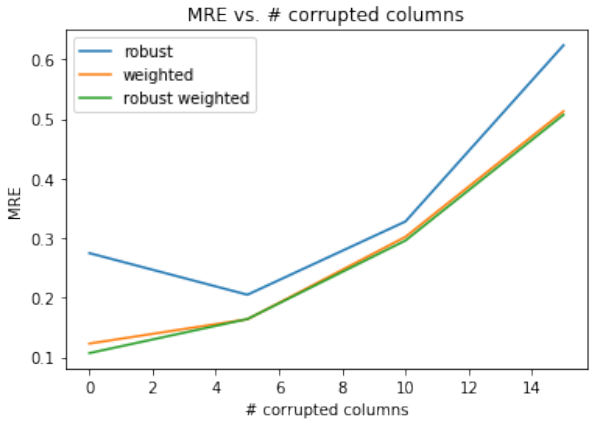

Figure 6: Mean recovery error of matrices at different corruption level in uniform observation

The matrix completion algorithm still attempts to converge to a low rank subspace with the corruption. Therefore, it is necessary that the algorithm be able to not only identify the columns, but also precisely recover their values as well, in order to correctly remove them from the low rank matrix. Figure 6 reinforces the inability to remove the corruption, as the unweighted minimization performs worse than the two weighted minimization.

4 Applications

4.1 Real world Applications

- *Video Surveillance*: A key motivating application of robust PCA is video analytics: the goal is to decompose a video (or an image) into a slowly changing background image and a sparse foreground image. Not only the background changes slowly, but the changes usually are dense (and not sparse, hence enforcing our incoherence assumption). Hence the background video matrix can be modelled as lying in a low-dimensional subspace of the original space. Since the foreground usually contains one or few moving objects, it is modelled as a sparse outlier.

- *Recommendation Systems*: It actually involves solving the robust Matrix Completion problem. Let's take the Netflix example: given n movies and d users, we denote the vector of true movie preferences for user t by ℓ_t, while the vector s_t contains any outlier entries for the same user. Assuming that user preferences are well-governed by only a few factors r_L, we can then correctly model $\mathbf{L}$ as low-rank. The outliers on the other hand occur because of corrupted entries entered by malicious users, incorrect ratings due to laziness, amongst others, and these are undeniably sparse. The

data matrix $\mathbf{X} = \mathbf{L} + \mathbf{S}$ also has unobserved entries because any user doesn't rate all the d movies. The goal is to recover the matrix $\mathbf{L}$ of true matrix preferences while at the same time being robust to the outliers.

5 Conclusion

In this study we investigated the behaviour of matrix completion algorithms in sub-optimal conditions. In particular, we considered cases where matrices have non-uniform entry observation pattern, cases where matrices have corrupted columns, and cases where matrices have high coherence structure. For each of these conditions, algorithms and adjustments to algorithms have been introduced to resolve the sub-optimal condition. Non-uniformity and coherence condition can be relaxed by the weighted nuclear norm minimization. As we discussed in this study, weighted nuclear norm minimization is a more flexible model because it allows each column and row to be penalized differently depending on how frequently they are observed. The corrupted/sparse column problem can be resolved by robust PCA, or robust matrix completion. We attempted to bring together these two ideas and develop a robust matrix completion algorithm that can perform robustly under non-uniform observation.

We conducted numerical experiments to verify these relaxations. For the weighted nuclear norm minimization, we showed that it outperforms the unweighted minimization in every setting. For robust matrix completion, while the algorithm is able to correctly identify the corrupted columns, it fails to capture the values of the entries, as the recovered matrix often has very small magnitude. In our experiment combining the two methods, however, the inability to remove the corruption in the low rank matrix becomes a major hurdle, as robustness adjustment offers no performance boost over the weighted methods. We wish to improve this in further studies by modifying the robust matrix completion algorithm to not only identify the corrupted columns, but also recover their values in order to remove the effect from the low rank component.

6 Future Works

The matrix completion problem has many moving parts. And as shown in this study the performance can depend heavily on the parameters such as penalty, frequency weight, and sparse percentage threshold. There are still many other parameters that we did not consider in this study. For example, how does the magnitude of the values of the corrupted entries affect its recovery? Could introducing adaptive thresholding improve the convergence of sparse component discovery? How does it perform when the sparse component is not column sparse but entry sparse?

Lastly, with the close association between the definition of local coherence and statistical leverage, it would be an interesting study to investigate their relationship. Tying together with matrix completion, using these information

to construct a more sophisticated weighing strategy may further improve the precision and efficiency of the algorithms.

7 Code

The code for the experiments can be found on github:
`https://github.coecis.cornell.edu/hs985/6520_Final`

References

[1] Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011.

[2] Yudong Chen, Huan Xu, Constantine Caramanis, and Sujay Sanghavi. Robust matrix completion and corrupted columns. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, ICML'11, pages 873–880, USA, 2011. Omnipress.

[3] Praneeth Netrapalli, Niranjan U N, Sujay Sanghavi, Animashree Anandkumar, and Prateek Jain. Non-convex robust pca. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1107–1115. Curran Associates, Inc., 2014.

[4] Nathan Srebro and Adi Shraibman. Rank, trace-norm and max-norm. In *COLT*, 2005.

[5] Xinyang Yi, Dohyung Park, Yudong Chen, and Constantine Caramanis. Fast algorithms for robust pca via gradient descent. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4152–4160. Curran Associates, Inc., 2016.