Convex and Non-Smooth Optimization
Spring, 2018
Project Report
Peter Will

Introduction

The Matrix Completion Problem consists of reconstructing the unknown or missing entries of a matrix from a relatively small set of its entries. This problem features prominently in the design of recommendation systems, where a company may wish to make personalized item recommendations to its users on the basis of historical user-item feedback, as in the well-publicized NetFlix Competition. Clearly, not all matrices can be reconstructed in this fashion. This is immediately apparent for full rank matrices, as such a matrix $\in \mathbb{R}^{m \times n}$ will have $m \times n$ degrees of freedom, and it is impossible to reconstruct it without sampling all $m \times n$ entries. On the other hand, if the matrix is of low rank then reconstruction from a partial sample becomes a tractable problem, as sampled entries "encode," so to speak, information about the full matrix. In 2009, Candés and Recht extended theoretical conditions under which perfect recovery of low-rank matrices from uniform random sampling is assured with high probability, based upon the "incoherence" of the matrix's singular vectors [2]. Among other things, the authors showed that a low rank matrix solution could be obtained via the convex relaxation:

$$\begin{aligned} \min \quad & ||X||_* \\ \text{s.t.} \quad & X_{ij} = M_{ij}, \quad (i,j) \in \Omega \end{aligned} \tag{1}$$

Where $||X||_*$ is the nuclear norm of the low-rank reconstruction matrix X and Ω is the set of indices over which the elements of X are required to match those of the original matrix M. An interesting variation of this problem is the noisy matrix completion problem.

$$\begin{aligned} \min \quad & ||X||_* \\ \text{s.t.} \quad & ||X_\Omega - M_\Omega||_F \leq \delta \end{aligned} \tag{2}$$

In 2, the sampling constraint is relaxed so that entries of the original matrix are only required to be "δ-close" to those of the low-rank reconstruction. This formulation of the matrix completion problem is of real-world importance, as there is often a source of error in the observed values of the original matrix. Alternatively, if the original matrix is not itself low-rank, one may still be interested in finding an approximate low-rank representation for it.

The convex formulation of the matrix completion problem shown in 1 and 2 has spurred much investigation into performant algorithmic solutions. Candes and Recht demonstrated that the problems could be cast as semi-definite programs, and are therefor able to be solved by general interior point methods. However, these methods scale poorly and are therefor unsuitable for application to large datasets. Consequently, in recent years many algorithms have been proposed. Specifically, there has been a resurgence of interest in Alternating Direction Method algorithms for the matrix completion problem owing to its decomposable structure and analytically solvable sub problems.

ADM for MCC

Recent papers [5], [8] have studied the application of ADM methods to the noisy matrix completion problem and their results are summarized below. In the following, the noisy version of the problem given in 2 is first rewritten in ADM form. Expressions for the augmented Lagrangian and resulting sub-problems are then derived. Finally, the analytical solutions to these sub-problems along with explicit ADM iterate-updates are produced.

We begin by introducing an auxiliary variable Y into 2 and writing the problem in ADM form as.

$$\begin{aligned} \min \quad & ||X||_* \\ \text{s.t.} \quad & X - Y = 0 \\ & X \in \mathbb{R}^{m \times x},\ Y \in \mathbf{B}_\delta \end{aligned} \tag{3}$$

where for $\delta \geq 0$

$$\mathbf{B}_\delta := \{U \in \mathbb{R}^{m \times n} : ||U_\Omega - M_\Omega||_F \leq \delta\} \tag{4}$$

represents the convex subspace of $\mathbb{R}^{m \times n}$ over which the relaxed sampling constraints are satisfied. The augmented Lagrangian of 3 is given by

$$\mathcal{L}(X, Y, Z, \beta) := ||X||_* - \langle Z, X - Y \rangle + \frac{\beta}{2}||X - Y||_F^2 \tag{5}$$

Where Z is the Lagrangian multiplier and $\beta \geq 0$ is the penalty parameter. At a high level, ADM works by alternating between minimizing 5 with respect to the variables X and Y. Because the objective function of 3 is decomposable and is constrained over a convex set, each of the resulting sub-problems can be solved analytically. The actual iterative scheme for ADM of 3 is as follows:

$$X^{k+1} = \arg\min_X \mathcal{L}(X^k, Y^k, Z^k, \beta) \tag{6a}$$

$$Y^{k+1} = \arg\min_Y \mathcal{L}(X^{k+1}, Y^k, Z^k, \beta) \tag{6b}$$

$$Z^{k+1} = Z^k - \beta\left(X^{k+1} - Y^{k+1}\right) \tag{6c}$$

Optimality conditions require that for 6a

$$0 \in \partial(||X^k||_*) - \left[Z^k - \beta(X^k - Y^k)\right] \tag{7}$$

Following the approach of [1], 7 can be solved via singular value shrinkage computation. More specifically, let

$$A^k = \frac{1}{\beta}Z^k + Y^k$$

And denote its singular value decomposition as

$$A^k = U^k \Sigma^k (V^k)^T$$

Using this definition, 7 can be rewritten as

$$0 \in \frac{1}{\beta}||X^k||_* + X^k - A^k \tag{8}$$

Which is satisfied when

$$X^k = U^k \hat{\Sigma}^k (V^k)^T \tag{9}$$

Where $\hat{\Sigma}^k$ is the matrix of singular values given by diag($\{(\sigma_i^k - \frac{1}{\beta})_+\}$), and U and V are the matrices of left and right singular vectors of A. For convenience, we can define this singular value shrinkage operator as

$$\mathcal{D}_{\frac{1}{\beta}}(A) = U(\Sigma - \frac{1}{\beta}I)_+(V)^T$$

Thus, an analytical solution to the subproblem given in 6a exists. It should be noted that it requires a singular value decomposition of the matrix X which generally requires a $\mathcal{O}(n^3)$ complexity cost. This computation ultimately ends up dominating the ADM's overall computational cost.

Turning now to the sub-problem of 6b, optimality conditions require that

$$\langle Y - Y^{k+1}, Y^{k+1} + \frac{1}{\beta} Z^k - X^k \rangle \geq 0, \; \forall Y \in \mathbf{B}_\delta \tag{10}$$

Consideration of this inequality shows that it is satisfied for all $Y \in \mathbf{B}_\delta$ if we chose Y^{k+1} to be equal to the projection of $X^k - \frac{1}{\beta} Z^k$ onto $\mathbf{B}_\delta$. Let $\mathcal{P}_{\mathbf{B}_\delta}$ denote such a projection operator, and let $B = X^k - \frac{1}{\beta} Z^k$. An explicit formula for $\mathcal{P}_{\mathbf{B}_\delta}$ can be derived by defining it as its own minimization problem, in which

$$\begin{aligned} \mathcal{P}_{\mathbf{B}_\delta}(B) = \arg\min_{X} \quad & ||X - B||_F^2 \\ \text{s.t.} \quad & ||X_\Omega - M_\Omega||_F \leq \delta \end{aligned} \tag{11}$$

The Lagrangian for the above minimization problem is given by

$$\mathcal{L}(X, \nu) = ||X - B||_F^2 + \nu(||X_\Omega - M_\Omega||_F - \delta)$$

and KKT conditions require that

$$\begin{aligned} X - B + \nu(X_\Omega - M_\Omega) &= 0 \\ \nu(||X_\Omega - M_\Omega||_F - \delta) &= 0 \\ ||X_\Omega - M_\Omega||_F &\leq \delta \\ \nu &\geq 0 \end{aligned}$$

It is easy to verify that the above system of equations is satisfied when

$$\begin{aligned} \nu &= \max\left\{ \frac{||B_\Omega - M_\Omega||_F}{\delta} - 1, 0 \right\} \\ X &= B + \frac{\nu}{\nu + 1}(B_\Omega - M_\Omega) \end{aligned}$$

The solution for X obtained defines the projection of B onto $\mathbf{B}_\delta$. This in turn provides an analytical solution for the sub-problem of 6b. The explicit ADM iterate updates are summarized below.

ADM Updates For the Noisy Matrix Completion Problem

$$X^{k+1} = \mathcal{D}_{\frac{1}{\beta}}(\frac{1}{\beta} Z^k + Y^k) \tag{12a}$$

$$Y^{k+1} = \mathcal{P}_{\mathbf{B}_\delta}(X^{k+1} - \frac{1}{\beta} Z^k) \tag{12b}$$

$$Z^{k+1} = Z^k - \beta\left(X^{k+1} - Y^{k+1}\right) \tag{12c}$$

It should be emphasized that the ability to derive closed form expressions for the subproblems of $6a$ and $6b$ is a result of the relative simplicity of the sampling operator used in the constraint of 2.

Numerical Experiment 1: ADM Vs. CVX SDPT3

The first experiment I ran compared the performance and results of my implementation of the ADM algorithm to CVX's interior point method. I tested both the noiseless and noisy cases given in 1 and 2 under a wide range of parameters. Data was generated for the test problem by a forming a a matrix $M = M_l M_r^T$, where $M_l \in \mathbb{R}^{m\times r}$ and $M_r \in \mathbb{R}^{n\times r}$ are matrices with entries drawn independently from $\mathcal{N}(0,1)$. The resulting matrix $M \in \mathbb{R}^{m\times n}$ is of rank r.

I use SR to denote the sampling ratio of the original matrix, with $\mathrm{SR} = \frac{p}{mn}$, where $p = |\Omega|$ is the number of sampled entries. Per [3] I measured the accuracy of the matrix completion by the relative error, defined as:

$$\mathrm{REer} = \frac{||X^k - M||_F}{||M||_F}$$

For a stopping criterion I measured the relative change in iterate matrix reconstruction, terminating the algorithm when

$$\frac{||X^{k+1} - X^k||_F}{||X^{k+1}||_F} \leq tol = 2 \times 10^{-4}$$

In the experiments that follow when using MATLAB's CVX package, I explicitly set the solver precision to 10^{-4} for more meaningful comparisons. Results for the noiseless problem are shown in Table 1, and indicate that in all instances the ADM algorithm produced reconstruction matrices with similar errors to CVX's SDPT3 solver. Since, many of the parameter sets selected for testing did not allow for exact reconstruction of the original matrix, small differences in the reconstruction error are to expected.

In all cases, the measured run time of the ADM algorithm was faster than that of SDPT3. The relative speedup was most pronounced for large matrices with high sampling ratios. This is a result of the ADM algorithm's runtime being dominated by the singular value decomposition required in the X subproblem, which scales as $\mathcal{O}(N^3)$. This computational complexity is independent of the number of elements sampled from the original matrix. Indeed, ADM actually preforms better with increasing sampling ratios as fewer ADM iterations are required to reached the stopping criterion. Conversely, the general interior point method used by SDPT3 scales with the number of constraints, and thus performs much worse at under large sample ratios. Figure 1 illustrates how problem size and sampling ratio impacts the relative speedup of the methods, defined by

$$\mathrm{Speedup} = \frac{\mathrm{CVX\ Time}}{\mathrm{ADM\ Time}}$$

Finally, Table 2 compares performance of the two solver algorithms on a noisy version of the matrix completion problem for varying orders of magnitude of noise. Again similar reconstruction errors are obtained for both methods. The results indicate that the runtime of the ADM does not increase in the presence of noise, unlike the SDPT3 solver.

Numerical Experiment 2: Using ADM to probe the boundary of recoverable matrices

Since the implemented ADM method proved to accurately reconstruct large matrices, I wanted to explore using it to experimentally verify the results of Candés and Recht's 2008 paper. In this paper the authors state the following theorem:

Theorem 1. *Let M be an $n_1 \times n_2$ matrix of rank r with entries sampled uniformly from the random orthogonal model. If m entries are observed from M, then there are constants C and c such that if,*

$$m \geq C n^{\frac{5}{4}} r \log n,$$

then the minimizer to the problem 1 is unique and equal to M with probability at least $1 - cn^{-3}$.

Table 1: Comparison of ADM and CVX SDPT3 for Noiseless problem

(N, r, SR)	ADM REer	ADM Time (secs)	CVX REer	CVX Time (secs)
(25, 10, 0.1)	1.00790	0.164	1.01480	0.230
(25, 10, 0.25)	0.93850	0.034	0.93880	0.230
(25, 10, 0.75)	0.23280	0.015	0.23270	0.430
(25, 20, 0.1)	1.00400	0.209	1.02940	0.230
(25, 20, 0.25)	0.90770	0.033	0.90800	0.250
(25, 20, 0.75)	0.43280	0.012	0.43280	0.420
(50, 10, 0.1)	0.98800	0.276	0.98730	0.460
(50, 10, 0.25)	0.73940	0.057	0.73940	0.840
(50, 10, 0.75)	0.00036	0.027	0.00044	2.190
(50, 20, 0.1)	1.03660	0.284	1.04020	0.450
(50, 20, 0.25)	0.88650	0.065	0.88660	0.790
(50, 20, 0.75)	0.16490	0.034	0.16510	3.380
(100, 10, 0.1)	0.93160	0.227	0.93190	2.000
(100, 10, 0.25)	0.44940	0.139	0.44960	7.570
(100, 10, 0.75)	0.00015	0.058	0.00001	30.120
(100, 20, 0.1)	0.97590	0.340	0.97610	1.000
(100, 20, 0.25)	0.72670	0.132	0.72670	4.100
(100, 20, 0.75)	0.00026	0.064	0.00002	34.920
(500, 10, 0.1)	0.07350	8.906	-	-
(500, 10, 0.25)	0.00025	2.637	-	-
(500, 10, 0.75)	0.00008	1.354	-	-
(500, 20, 0.1)	0.60850	4.164	-	-
(500, 20, 0.25)	0.00033	2.657	-	-
(500, 20, 0.75)	0.00013	1.354	-	-
(1000, 10, 0.1)	0.00042	29.552	-	-
(1000, 10, 0.25)	0.00022	19.935	-	-
(1000, 10, 0.75)	0.00007	7.681	-	-
(1000, 20, 0.1)	0.02690	51.298	-	-
(1000, 20, 0.25)	0.00028	17.393	-	-
(1000, 20, 0.75)	0.00011	8.232	-	-

Results are shown for groupings of (N, r, SR) for random matrices generated in $\mathbb{R}^{N\times N}$ of Rank r

Table 2: Comparison of ADM and CVX SDPT3 for Noisy problem

(N, r, SR, δ)	ADM REer	ADM Time (secs)	CVX REer	CVX Time (secs)
(100, 10, 0.1, 0.1)	0.93150	0.228	0.93150	1.120
(100, 10, 0.1, 1)	0.93250	0.248	0.93240	1.150
(100, 10, 0.1, 10)	0.79220	0.325	0.79200	1.210
(100, 10, 0.25, 0.1)	0.45080	0.134	0.45140	4.350
(100, 10, 0.25, 1)	0.51560	0.133	0.51580	4.670
(100, 10, 0.25, 10)	0.61540	0.134	0.61540	5.140
(100, 10, 0.75, 0.1)	0.00840	0.084	0.00670	65.710
(100, 10, 0.75, 1)	0.06490	0.151	0.06500	101.710
(100, 10, 0.75, 10)	0.29690	0.057	0.29720	121.080

Figure 1: Relative Speedup of ADM:CVX, for a random matrix of Rank = 10

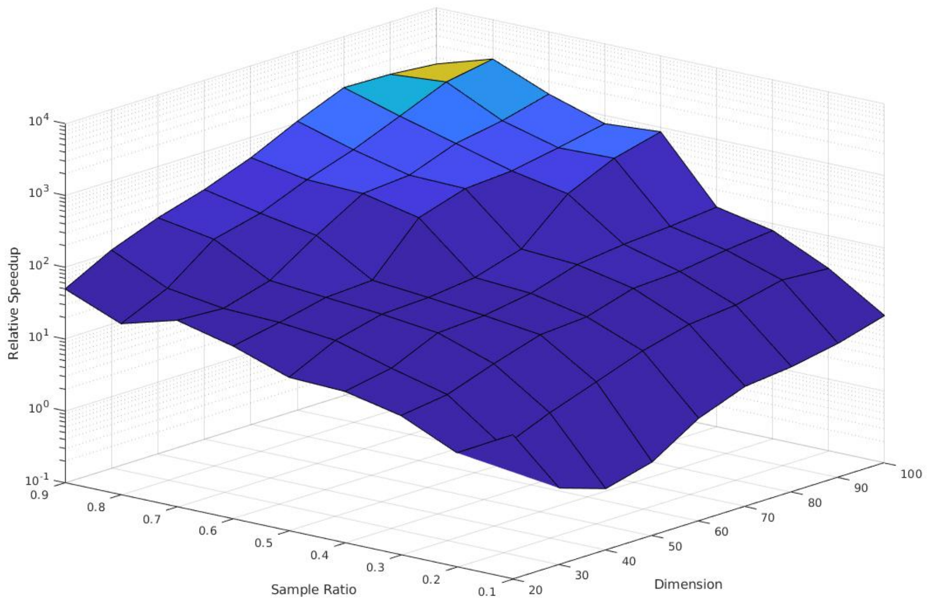
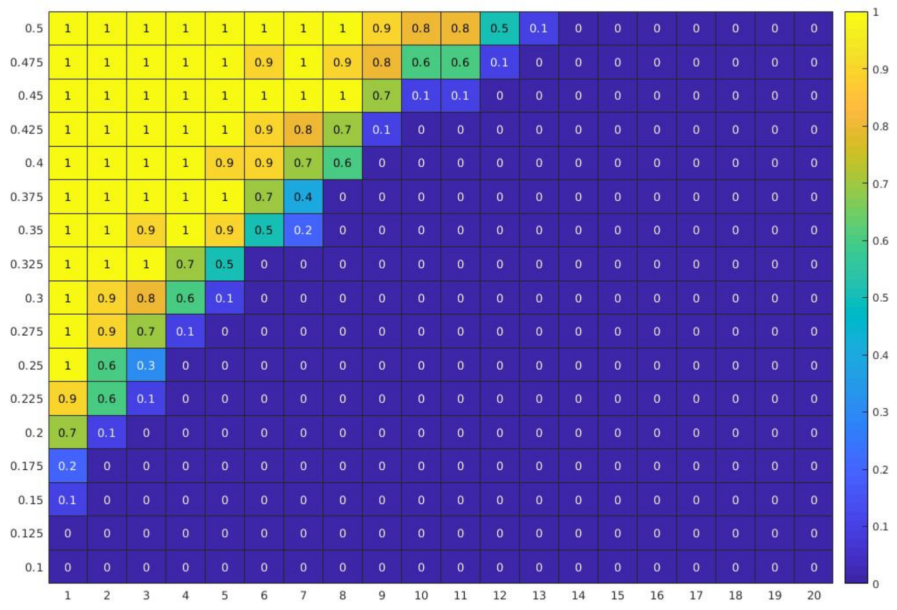


Figure 2: ADM recovery of low rank solutions. The x-axis shows the rank of the generated data matrix. The y-axis shows the sampling ratio. The cell-value represents the proportion of trials that resulted in reconstruction errors less than the threshold of 10^{-3}



Since the setup in the first experiment fits the assumptions of the theorem it is of interest to see if experimental evidence for this theory can be observed. In this experiment, the matrix dimensions were fixed at 100×100, and 10 independent trials were performed for different rank and sampling ratio parameter sets. As Figure 2 shows, there is a clear boundary in the nature of the reconstruction error as rank and sampling ratio change. More specifically, exact matrix recovery occurs under relatively high sampling ratios and low rank parameter pairings.

Extending the Model to a new class of problems

So far, the models considered in 1 and 2 have focused on matrix completion from a partial sampling of elements. However, the ADM algorithm and associated subproblem solutions extend naturally to another interesting problem. The Low Rank Recovery problem attempts to find the nearest low-rank approximation of a given matrix. A special form of it is given in 13 below:

$$\begin{aligned} \min_{A,E} \quad & ||A||_* + \tau||E||_1 \\ \text{s.t.} \quad & ||M - A - E||_F \leq \delta \end{aligned} \tag{13}$$

Here, once again, we seek a low rank approximation of a matrix via the use of the nuclear norm. Unlike the problem posed in 1 and 2, however, we assume full knowledge of the original matrix A. Additionally, we assume that the entires of A are subject to two types of corruption. The first is a low magnitude Gaussian white noise, assumed to effect all entries of A. This is introduced by means of the inequality constraint. The second is an arbitrarily large, sparse source of error. This "impulse error" enters the problem as the matrix E, and is minimized directly in the objective function.

This model captures the so-called Robust Principal Component Analysis (PCA) problem. PCA is a fundamental operation in data analysis and is used extensively in statistics, computer vision, and machine learning applications. Unfortunately, PCA can fail badly when the data contains outliers or badly corrupted elements. The minimization problem 13 presents an approach that produces a low-rank representation of the data which is robust in the presence of these sorts of abnormalities.

To reformulate the problem in ADM form, I follow the approach of [6] and introduce a variable substitution $Y = M - A - E$, is used. The new problem becomes:

$$\begin{aligned} \min_{A,E,Y} \quad & ||A||_* + \tau||E||_1 \\ \text{s.t.} \quad & A + E + A = M, \\ \text{s.t.} \quad & Y \in \mathbf{B} := \{Y \in \mathbb{R}^{m\times n} \mid ||Y||_F \leq \delta\} \end{aligned} \tag{14}$$

Note that while the convex constraint on Y no longer makes use of a sampling operator, 14 bares a strong resemble to 3. The augmented Lagrangian of 14 is given by:

$$\mathcal{L}(A, E, Y, Z, \beta) := ||A||_* + \tau||E||_1 - \langle Z, A + E + Y - M\rangle + \frac{\beta}{2}||A + E + Y - M||_F^2 \tag{15}$$

And, the ADM subproblems now take the form

$$Y^{k+1} = \arg \min_{Y\in\mathbf{B}} \frac{\beta}{2}||Y + A^k + E^k - \frac{1}{\beta}Z^k - M||_F^2 \tag{16a}$$

$$E^{k+1} = \arg \min_{E} \tau||E||_1 + \frac{\beta}{2}||E + A^k + Y^{k+1} - \frac{1}{\beta}Z^k - M||_F^2 \tag{16b}$$

$$A^{k+1} = \arg \min_{A} ||A||_* + \frac{\beta}{2}||A + E^{k+1} + Y^{k+1} - \frac{1}{\beta}Z^k - M||_F^2 \tag{16c}$$

$$Z^{k+1} = Z^k - \beta\left(A^{k+1} + E^{k+1} + Y^{k+1} - M\right) \tag{16d}$$

In solving these subproblems we note that the updates for Y and A are nearly identical to the solutions produced for subproblems 6b and 6a, respectively. Indeed, the update for Y is given by the same projection operator derived in 11, however, the sampling operator is no longer used. Similarly, an analytical solution for 16c can be written in terms of the shrinkage operator defined in 9. For the subproblem of 16b we make use of the following lemma found in [5].

Lemma 2. *For $\mu > 0$ and $\mathbf{T} \in \mathbb{R}^{m\times n}$, the solution of the follwoing problem*

$$\min_S \mu||S||_1 + \frac{1}{2}||S - \mathbf{T}||_F^2,$$

is given by $\mathcal{S}_\mu(\mathbf{T} \in \mathbb{R}^{m\times n}$, which is defined component-wise as

$$(\mathcal{S}_u(\mathbf{T}))_{ij} := \max\{abs(\mathbf{T}_{ij}) - \mu, 0\} \cdot sign(\mathbf{T}_{ij}).$$

Thus, as before, the ADM iterate updates all have analytical solutions. These are presented below:

ADM updates for Noisy Low Rank Recovery

$$Y^{k+1} = \mathcal{P}_{\mathbf{B}_\delta}(\frac{1}{\beta}Z^k + M - A^k - E^k) \tag{17a}$$

$$E^{k+1} = \mathcal{S}_{\tau/\beta}(\frac{1}{\beta}Z^k + M - A^k - Y^{k+1}) \tag{17b}$$

$$A^{k+1} = \mathcal{D}_{\frac{1}{\beta}}(\frac{1}{\beta}Z^k + M - Y^{k+1} - E^{k+1}) \tag{17c}$$

$$Z^{k+1} = Z^k - \beta\left(A^{k+1} + E^{k+1} + Y^{k+1} - M\right) \tag{17d}$$

Numerical Experiment 3: Low Rank Recovery in the presence of noise and sparse impulse errors

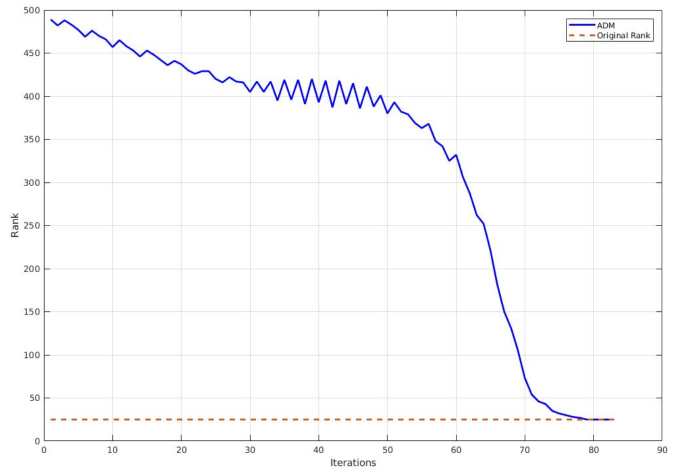
I tested the efficacy of the ADM implementation for the problem given in 13 by examining its performance on a matrix $A = M + S + E$. Here, M represents the original low rank data we wish to recover. It was created by taking the matrix product of M_l and M_r^T, where $M_l \in \mathbb{R}^{m\times r}$ and $M_r \in \mathbb{R}^{n\times r}$ are matrices with entries drawn independently from $\mathcal{N}(0,1)$. The matrix $S \in \mathbb{R}^{n\times m}$ had entries drawn from $\mathcal{N}(0,1)$, and denotes the Gaussian noise effecting all entries of M. Finally, the support for the sparse matrix E was chosen uniformly at random. It's non-zero entries were given values drawn uniformly from the interval [-500,500]. Following the approach of Tao & Yuan I set $\tau = \frac{1}{\sqrt{n}}$. Figure 3 shows how the rank of the recovered matrix converges to the rank of the original matrix as the ADM algorithm iterates.

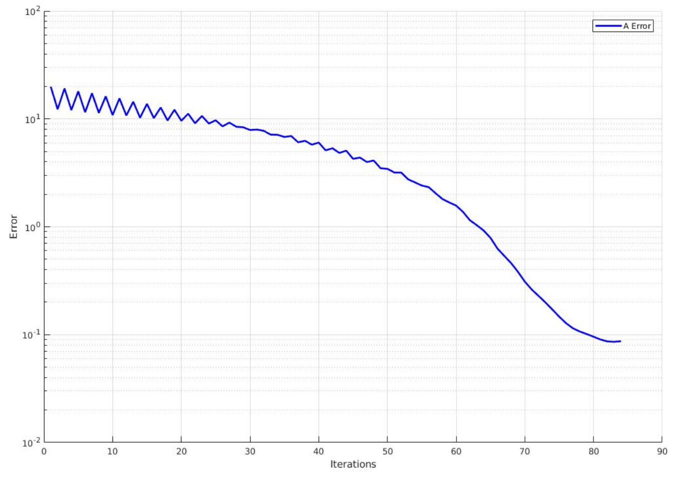
Conclusion

This report has surveyed some of the recent work done to solve the Matrix Completion and Noisy Low Rank Recovery problems. The ADM algorithm is especially well suited to these tasks as they feature separable objective functions over convex constraints. The resulting sub-problems are simple enough that they can be solved analytically. The dominate cost for both the ADM matrix completion algorithm and the ADM low rank recovery algorithm is a singular value decomposition. This scales with the problem size as $\mathcal{O}(n^3)$ and consequently, the algorithms vastly outperformed the general SDPT3 solver.

Further acceleration of these methods is possible. While in this report and accompanying MATLAB code I performed a full singular value decomposition of the given matrix, as authors Lin et al. point out, only a partial decomposition is required. Using the MATLAB library PROPACK and rank prediction it is possible to reduce the decomposition cost to $\mathcal{O}(rN^2)$. Additionally, many problems faced in the real world do not lend themselves to such tidy analytical solutions. Given more time, I would like to analyze some of the recent work that has gone into linearizing more complex operators for the Matrix Completion and Low Rank Recovery problems.

Figure 3: ADM recovery of low rank solutions. (TOP) The rank of the recovered matrix M^* at each iteration is plotted for an example matrix $A \in \mathbb{R}^{500\times 500}$ with rank = 25. $\mathrm{nnz}(E) = 12500$. (BOTTOM) The reconstruction error of the recovered matrix.





References

[1] Cai, Jian-Feng, Emmanuel J. Candés, and Zuowei Shen. *A singular value thresholding algorithm for matrix completion.* SIAM Journal on Optimization 20, no. 4 (2010): 1956-1982.

[2] Candés, Emmanuel J., and Benjamin Recht. *Exact matrix completion via convex optimization.* Foundations of Computational mathematics 9, no. 6 (2009): 717.

[3] Chen, Caihua, Bingsheng He, and Xiaoming Yuan. *Matrix completion via an alternating direction method.* IMA Journal of Numerical Analysis 32, no. 1 (2012): 227-245.

[4] Lin, Zhouchen, Risheng Liu, and Zhixun Su. *Linearized alternating direction method with adaptive penalty for low-rank representation.* Advances in neural information processing systems, pp. 612-620. 2011

[5] Tao, Min, and Xiaoming Yuan. *Recovering low-rank and sparse components of matrices from incomplete and noisy observations.* SIAM Journal on Optimization 21, no. 1 (2011): 57-81.

[6] Wright, John, Arvind Ganesh, Shankar Rao, Yigang Peng, and Yi Ma. *Robust principal component analysis: Exact recovery of corrupted low-rank matrices via convex optimization.* Advances in neural information processing systems, pp. 2080-2088. 2009.

[7] Xiaoming Yuan and Junfeng Yang. *Sparse and Low-Rank Matrix Decomposition Via Alternating Direction Method*. preprint, 12, 2..

[8] Yang, Junfeng, and Xiaoming Yuan. *Linearized augmented Lagrangian and alternating direction methods for nuclear norm minimization.* Mathematics of computation 82, no. 281 (2013): 301-329.

Matlab Code

ADM_MC.m

This script contained my implementation of the ADM solution for the noisy matrix completion problem used for the first two numerical experiments.

```
function [X_hat, iter, time, opt_val] = ADM(M, inds, delta)
tic
[m,n] = size(M);
X = randn(size(M));
X(inds) = M(inds);
Z = randn(size(M));
tol = 10^-4;
stop_criteria = inf;
iter = 0;
beta = 2.5 / (m*n)^.5;
gamma = 1.6;
while(stop_criteria > tol)
    iter = iter +1;
    B = X - (1/beta)*Z;
    Y = B;
    temp = B(inds) - M(inds);
    Y(inds) = (min(delta / norm(temp, 'fro'), 1) - 1)*temp +B(inds
        );
    [U,S,V] = svd(Y+(1/beta)*Z, 'econ');
    S_hat = diag(max(diag(S) - (1/beta),0));
    X_new = U*S_hat*V';
    stop_criteria = REER(X, X_new);
    X = X_new;
    Z = Z - gamma*beta*(X-Y);
    %fprintf('Iter: %d, Err: %f, Tol: %f\n',iter, REER(X,M),
        stop_criteria);
end
time = toc;
[u,s,v] = svd(X);
opt_val = sum(sum(s));
X_hat = X;
```

gen_data.m

This script contained a helper function to generate data for the first two numerical experiments.

```
function [M,inds,delta] = gen_data(m,n,r,SR, noise)
Ml = randn(m,r);
Mr = randn(r,n);
M = Ml*Mr;
assert(rank(M) == r);
k = ceil(SR*m*n);
```

```
inds = randsample(1:(m*n),k,false);
S = noise*rand(m,n);
M = M+S;
delta = .1*norm(S(inds),'fro');
```

cvx_solver.m

This script contained my implementation of CVX's SDPT3 solver for comparison against the ADM method in the first numerical experiment.

```
function X = cvx_solver(M, inds, delta)

[m, n] = size(M);
[~, k] = size(inds);
cvx_begin
    cvx_precision(10^-4);
    variable X(m,n);
    minimize norm_nuc(X)
    subject to
        norm(X(inds) - M(inds), 'fro') <= delta;
cvx_end

end
```

ADM_LRR.m

This script contained my implementation of the ADM solution for the Low Rank Recovery problem of the third numerical experiment.

```
function [A, iter, time] = ADM(M, M_star, E_star, delta, tau)
tic
[m,n] = size(M);
A = M;
Z = randn(size(M));
E = randn(size(M));

tol = 10^-4;
stop_criteria_lr = inf;
stop_criteria_er = inf;
iter = 0;
[m, n] = size(M);
beta = .2*(m*n) / sum(sum(abs(M)));
a_errors = [];
e_errors = [];
ranks = [];
while((stop_criteria_lr > tol) || (stop_criteria_er > tol))
    iter = iter +1;
    N = (1/beta)*Z + M - A - E;
    Y = (min(norm(N,'fro'),delta) / norm(N,'fro')) * N;
    E_new = Soft((1/beta)*Z + M  - A - Y, tau / beta);
    A_new = D((1/beta)*Z + M - Y - E, 1/beta);
    Z_new = Z - beta*(A+E+Y-M);
    stop_criteria_lr = beta*norm(A-A_new,'fro') / (norm(A,'fro') +
        1) ;
    stop_criteria_er = beta*norm(E-E_new,'fro') / (norm(E,'fro') +
        1) ;
    A = A_new;
    E = E_new;
    Z = Z_new;
    fprintf('Iter: %d, Err: %f, stop_criteria_lr: %f,
        stop_criteria_er: %f\n'...
        ,iter, REER(A,M_star), stop_criteria_lr, stop_criteria_er)
            ;
    a_errors = [a_errors, REER(A,M_star)];
    e_errors = [e_errors, REER(E,E_star)];
    ranks = [ranks, rank(A)];
end
time = toc;
figure(1)
hold on
plot(1:iter, a_errors)
plot(1:iter, e_errors)
hold off
figure(2)
plot(1:iter, ranks)
```

gen_data_lrr.m

This script contained a helper function to generate data for the third numerical experiment.

```
function [M_star,E,M,delta] = gen_data(m,n,r,SR,noise)
Ml = randn(m,r);
Mr = randn(r,n);
M_star = Ml*Mr;
assert(rank(M_star) == r);
E = sprand(m,n,SR);
E(E > 0)  = 1000*(rand([nnz(E),1])-.5);

S = noise*rand(m,n);
M = M_star+E+S;
d = min(m,n);
delta = (d + ((8*d)^.5))*.5;
```